

MAGAZINE

BSD

FOR NOVICE AND ADVANCED USERS

INTERVIEW ISSUE!

DEB GOODKIN, EXECUTIVE DIRECTOR FOR THE **FREEBSD** FOUNDATION

SHAWN WEBB, FOUNDER OF **HARDENEDBSD** PROJECT

HENNING BRAUER, CEO OF **BS WEB SERVICES GMBH**

MATHILDE FFRENCH, FOUNDER OF **ECHINOPSII**

VMWARE STORAGE DRS

TO THE RESCUE FOR INTEGRATING **TRUENAS** DATA STORAGE

**THE VOICEMAIL SCAMMERS NEVER GOT
PAST OUR OPENBSD GREYLISTING**

SSHLOCK – SSH ON STEROIDS!

VOL 10 NO 12

ISSUE 11/2016 (88)

1898-9144

EDITORS' WORD

Dear Readers,

Thank you so much for your patience this issue.

We hope you had an amazing holiday break with your friends and family. New Year's Eve is just around the corner, so have an amazing night and a Happy New Year. Hopefully it will be better than 2016, which has been tough to all of us as World Citizens. But we are not here to talk about politics, celebrities or social issues. We are here to talk about BSD... a lot ;) This issue is full of interviews we have been working on for the past couple of months and managed to complete them all in the same time.

First, though, we will begin with the news section and a great article by Franck Porcher. Franck has been writing an article for our security issue and shared with us "SSHLock – SSH on Steroids!" Grab a cup of coffee, there is a lot to read.

Next we have "The Voicemail Scammers Never Got Past Our OpenBSD Greylisting" by Peter Hansteen, another article with a security-related subject, this time on OpenBSD.

Now, let's move on to the main section of this issue: interviews! We will begin with a great interview with Deb Goodkin, Executive Director of the FreeBSD Foundation. This interview has been very close to our hearts because the FreeBSD Foundation was waiting to reach their financial goal, and they still hadn't met it at the time of the interview.

The next interview you will read is with Shawn Webb, who we had an interview with around two years ago as well. It's been great to talk to him again, as he is so devoted to Open Source projects.

Our next interview is with Henning Brauer, who has been an OpenBSD developer for 14 years and CEO of BS Web Services for 20 years now. So, if you haven't heard about his company, this interview is a must read for you.

We are so happy to have another woman as our interviewee in this issue, Mathilde Ffrench, founder of echinop-sii. Mathilde shared a lot of details with us about their project, Ariane, a mapping automation framework.

iXsystems shared an article by Brad Meyer, Technical Marketing Engineer, called "VMware Storage DRS to the Rescue for Integrating TrueNAS Data Storage," and we know you will love it.

The cherry on top of this issue is Rob's Column and his thoughts about changes in the IT industry within the last 20 years. Rob, thank you and all the best for you from BSD Mag Team! We hope you will get better soon!

Dear Readers, all the best for you as well! Have a happy, favorable and peaceful New Year!

Marta & BSD Team

MAGAZINE **BSD**

Editor in Chief:

Marta Ziemianowicz

marta.ziemianowicz@software.com.pl

Contributing:

Franck Porcher, Peter Hansteen, Brad Meyer, Mathilde Ffrench, Henning Brauer, Shawn Webb, Deb Goodkin and Rob Somerville.

Top Betatesters & Proofreaders:

Denise Ebery, Eric Geissinger, Luca Ferrari, Imad Soltani, Olaoluwa Omo-kanwaye, Radjis Mahangoe, Mani Kanth and Mark VonFange.

Special Thanks:

Annie Zhang

Denise Ebery

DTP:

Marta Ziemianowicz

Senior Consultant/Publisher:

Paweł Marciniak

pawel@software.com.pl

CEO:

Joanna Kretowicz

joanna.kretowicz@software.com.pl

Publisher:

Hakin9 Media SK 02-676 Warsaw, Poland Postepu 17D Poland worldwide
publishing editors@bsdmag.org www.bsdmag.org

Hakin9 Media SK is looking for partners from all over the world. If you are interested in cooperation with us, please contact us via e-mail:
editors@bsdmag.org.

All trademarks presented in the magazine were used only for informative purposes. All rights to trademarks presented in the magazine are reserved by the companies which own them.

CONTENTS

News

BSD World Monthly News 4

by Marta Ziemianowicz

This column presents the latest news coverage of events, product releases and trending topics.

Security

SSHLock – SSH on Steroids! 13

by Franck Porcher, PhD

Whether locally, through an attached terminal, or remotely via SSH, standard access to computers has not evolved much in the last 40 years in spite of continuous technological advances. They still heavily rely on the traditional password-based authentication mechanism which, acting as the unique “protective door” between the world and the computer's valuable vault, most unfortunately represents a single *point of failure*.

OpenBSD

The Voicemail Scammers Never Got Past Our OpenBSD Greylisting 71

by Peter Hansteen

We usually don't see much of the scammy spam and malware, but when we went looking for them, we found a campaign where our OpenBSD greylisting setup was 100% effective in stopping the miscreants' messages.

Interview

Interview with Deb Goodkin 75

by Marta Ziemianowicz, Marta Strzelec & Marta Si-enicka

Interview with Shawn Webb 81

by Marta Ziemianowicz, Marta Strzelec & Marta Si-enicka

Interview with Henning Brauer 85

by Marta Ziemianowicz, Marta Strzelec & Marta Si-enicka

Interview with Mathilde Ffrench 90

by Marta Ziemianowicz, Marta Strzelec & Marta Si-enicka

FreeNAS

VMware Storage DRS to the Rescue for Integrating TrueNAS Data Storage 102

by Brad Meyer

Integrating a new data storage system into an existing VMware environment can challenge even the most experienced administrator. How do you integrate the new storage into VMware while minimizing operational impact? Which VMs should move to the new storage? How do you ensure that you do not overload one storage system over another? How do I retire an older storage system without impacting my VMware environment? How much time does it take to manage these decisions? VMware Storage DRS is the answer.

Rob's Column 108

by Rob Somerville

More than 30 years have passed since he first typed at a computer keyboard. Rob Somerville looks back over the major changes in the IT industry and enquires what will be the next revolution on the horizon.

Simplify your Data Center

Meet TrueRack™ — A powerfully flexible rack-scale architecture that takes the guesswork out of building large scale data center applications.



- Converged Infrastructure
- Customizable for Virtualization, Big Data, Cloud & Hyperscale
- Up to 70% Lower TCO Than AWS
- Scalable and Repeatable Deployments

For more information on TrueRack, visit **iXsystems.com/TrueRack** today

BSD Certification

The BSD Certification Group Inc. (BSDCG) is a non-profit organization committed to creating and maintaining a global certification standard for system administration on BSD based operating systems.

? WHAT CERTIFICATIONS ARE AVAILABLE?

BSDA: Entry-level certification suited for candidates with a general Unix background and at least six months of experience with BSD systems.

BDSP: Advanced certification for senior system administrators with at least three years of experience on BSD systems. Successful BDSP candidates are able to demonstrate strong to expert skills in BSD Unix system administration.

✓ WHERE CAN I GET CERTIFIED?

We're pleased to announce that after 7 months of negotiations and the work required to make the exam available in a computer based format, that the BSDA exam is now available at several hundred testing centers around the world. Paper based BSDA exams cost \$75 USD. Computer based BSDA exams cost \$150 USD. The price of the BDSP exams are yet to be determined.

Payments are made through our registration website:
<https://register.bsdcertification.org/register/payment>

i WHERE CAN I GET MORE INFORMATION?

More information and links to our mailing lists, LinkedIn groups, and Facebook group are available at our website:
<http://www.bsdcertification.org>

Registration for upcoming exam events is available at our registration website:
<https://register.bsdcertification.org/register/get-a-bsdcg-id>

DragonFlyBSD's i915 Intel DRM Driver Updated To Match Linux 4.5

DragonFlyBSD and, in particular, Francois Tigot continue to do a good job at continuing to update their Intel "i915" Direct Rendering Manager driver to catch-up with what's offered by the mainline Linux kernel with their official Intel DRM driver.

The other week, Tigot re-based the drm/i915 DRM code against what's found in the Linux 4.5 kernel. This update changed around 10,000 lines of driver code within the DragonFlyBSD kernel. By this upgrade, the DragonFlyBSD Intel graphics driver has better Skylake and Broxton hardware support, initial Kaby Lake support, and many bug fixes, per this Git message.

Meanwhile, Linux 4.9.0 is set to be released this weekend and there are a lot of i915 changes from Linux 4.5 to 4.9, but still, the DragonFlyBSD (and overall BSD state on most distributions) continues advancing with their open-source graphics driver support and still much better off than a few years back when their DRM driver support was much more rudimentary. Today, it's at least very much usable assuming you aren't on the very recent hardware or affected by a recently fixed Linux kernel DRM bug.

http://www.phoronix.com/scan.php?page=news_item&px=Intel-DRM-DFly-Linux-4.5

NetApp Amazes (but iXsystems Disrupts)



NetApp is amazing! Yes, you heard me correctly. NetApp is indeed an amazing company. They were pioneers in networked file storage, made many great products over the years, and did a fantastic job of hiring great people to build the company's culture. These were the ingredients of the secret sauce that made NetApp special. So why would I ever leave NetApp to join iXsystems?

How do you leave giant, proprietary Disneyland to test the waters with a comparatively smaller, Open Source "upstart" (of 20 years and counting, by the way)? While this might puzzle some, the answer for me was pretty simple. Disruption. And yes, I know what you're thinking right now, 'Disruption? That's original.' Like you, I understand every company on the market is claiming they are disrupting the storage, network, or compute industry on a daily basis. The onslaught of press releases promising 'The next great thing!' is so common we've nearly become desensitized to the mere thought of 'disruption.' Only in the ego-driven tech industry could companies ever believe that they're changing the world with each dot release. Well, I can offer some relief here because the beautiful thing about the disruption I'm speaking of is not necessarily a technological breakthrough but a way of doing business with iXsystems.

iXsystems is gaining customers, some of whom are parting ways with NetApp and EMC (now Dell EMC) at such a rapid pace because we offer a highly flexible enterprise-grade solution that meets complex storage requirements without breaking the bank, giving up features, or sacrificing enterprise support. We do this by being the first storage company in the world that Open Sourced our storage operating system, FreeNAS, and it is now the world's leading Open Source Software Defined Storage OS with almost 9 million downloads worldwide. This model breeds open collaboration and feedback from a strong community of users and contributors to help develop business-ready storage software. Releasing FreeNAS into the wild allows us to also provide you with a battle-tested enterprise product in TrueNAS that provides more features and performance per dollar and continues to surpass expectations year after year.

The best part is we dare you to try FreeNAS because we know that once you do, you too will be asking why am I paying six or seven figures for a logo on a bezel? I left NetApp because I get it. Look, if you are in the 5% of storage projects that require a billion IOPS and limitless scale-out capability we have no problem letting you know you probably need to speak with NetApp or EMC. But more than likely, you probably fit within the majority of IT projects that could benefit from a fully unified hybrid storage system that gives you the flexibility to handle today's workload and the unforeseen requirements of the future.



The Open Source business model makes sense to me. Growing up near the birthplace of RedHat in Raleigh, North Carolina showed me how Open Source collaboration can change an industry firsthand and the tide is turning as more and more Open Source solutions enter production environments. Does this mean everyone is going to jump ship today? No, not at all, but I saw how iXsystems was still growing in a mostly flat storage market and realized that it was doing something special. iXsystems is disrupting how people acquire and use storage. That's why I left NetApp, and if you think I'm crazy why don't you give FreeNAS a try?

Patrick Bullock, Channel Account Manager

<https://www.ixsystems.com/blog/netapp-amazes-ixsystems-disrupts/>

NASA & FreeBSD: Building Cost-Effective 100-Gbps Firewalls for HPC



In this article, NASA discusses using FreeBSD servers for their Climate Simulation project. FreeBSD was chosen as a cost effective solution that also provided ample security and network reliability.

Overview

The continuous growth of the NASA Center for Climate Simulation (NCCS) requires providing high-performance security tools and enhancing the network capacity. In order to support the requirements of emerging services, including the Advanced Data Analytics Platform (ADAPT)

private cloud, the NCCS security team has proposed an architecture to provide extremely cost-effective 100-gigabit-per-second (Gbps) firewalls.

Project Details

The aim of this project is to create a commodity-based platform that can process enough packets per second (pps) to sustain a 100-Gbps workload within the NCCS computational environment. The test domain consists of several existing systems within the NCCS, including switches (Dell S4084), routers (Dell R530s), servers (Dell R420s, and C6100s), and host card adapters (10-Gbps Mellanox ConnectX2 and Intel 8259 x Ethernet cards).

Original article: <https://www.nasa.gov/SC16/demos/demo9.html>

<https://www.freebsdnews.com/2016/12/09/building-cost-effective-100-gbps-firewalls-hpc/>

Browsix: Now Run A Unix-like Open Source Operating System In Your Browser

A team of developers from the University of Massachusetts, Amherst, has created a Unix-like operating system for your web browser. It uses a JavaScript-based kernel and extends the JS runtimes for C, C++, Go, and Node.js programs. It also comes with a POSIX-like shell.

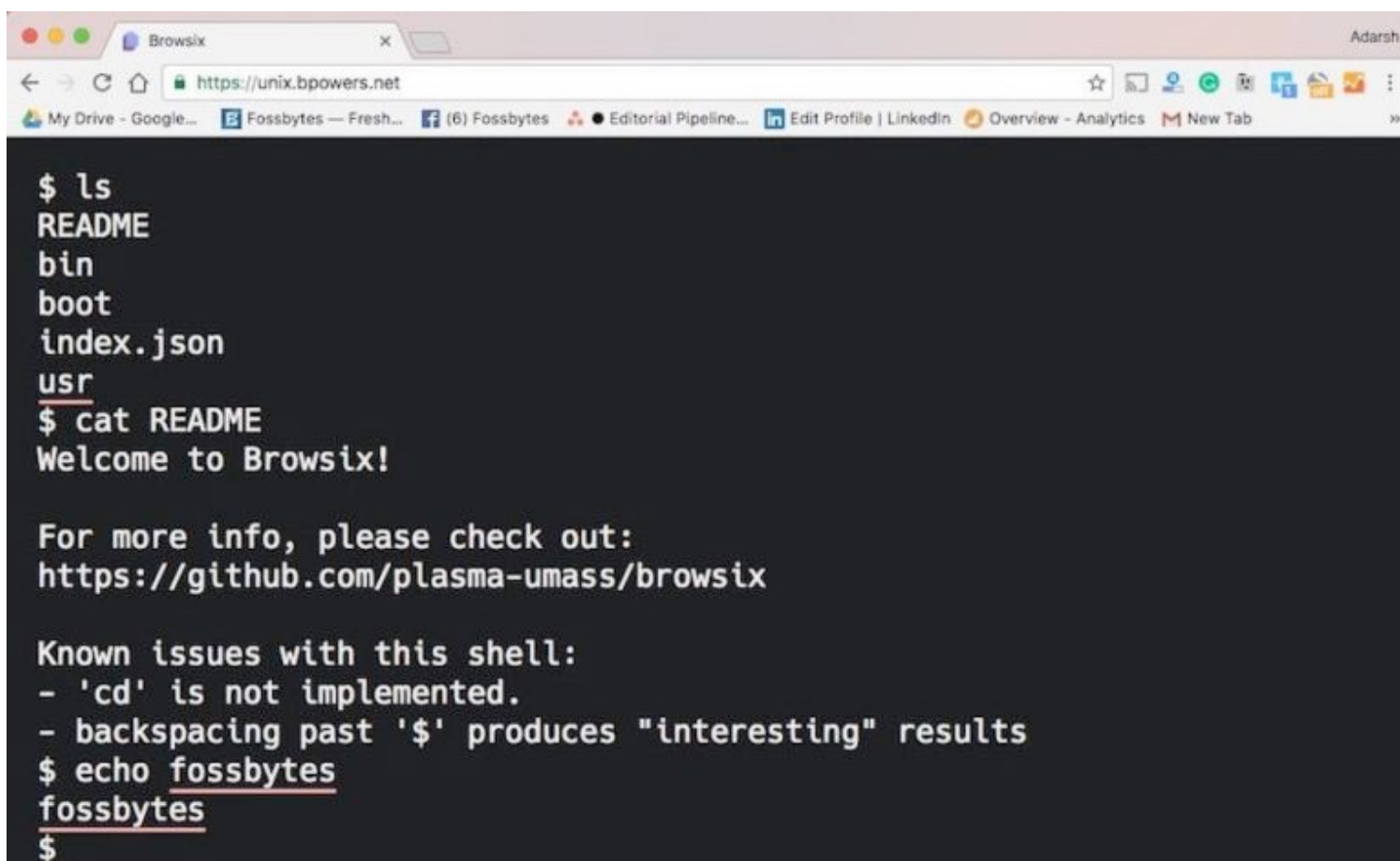
The modern web browsers are great for playing videos, reading blogs, or even building user interfaces. But they fail if you wish to use them as a platform to code-heavy applications. As the applications depend on standard OS APIs, something browsers don't support, compiling code to JS doesn't work.

On the other hand, operating systems, including Unix, allow the user to build applications with ease. To overcome this limitation of web browsers, Browsix has been created as a research project from the PLASMA lab at the University of Massachusetts, Amherst.

What is Browsix? How does it work?

Browsix consists of two core parts; a kernel written in TypeScript and extended runtimes for C, C++, Go, and Node.js. Let's tell you more about it.

Browsix is a JavaScript-only framework that makes the core Unix features available to web apps



```
$ ls
README
bin
boot
index.json
usr
$ cat README
Welcome to Browsix!

For more info, please check out:
https://github.com/plasma-umass/browsix

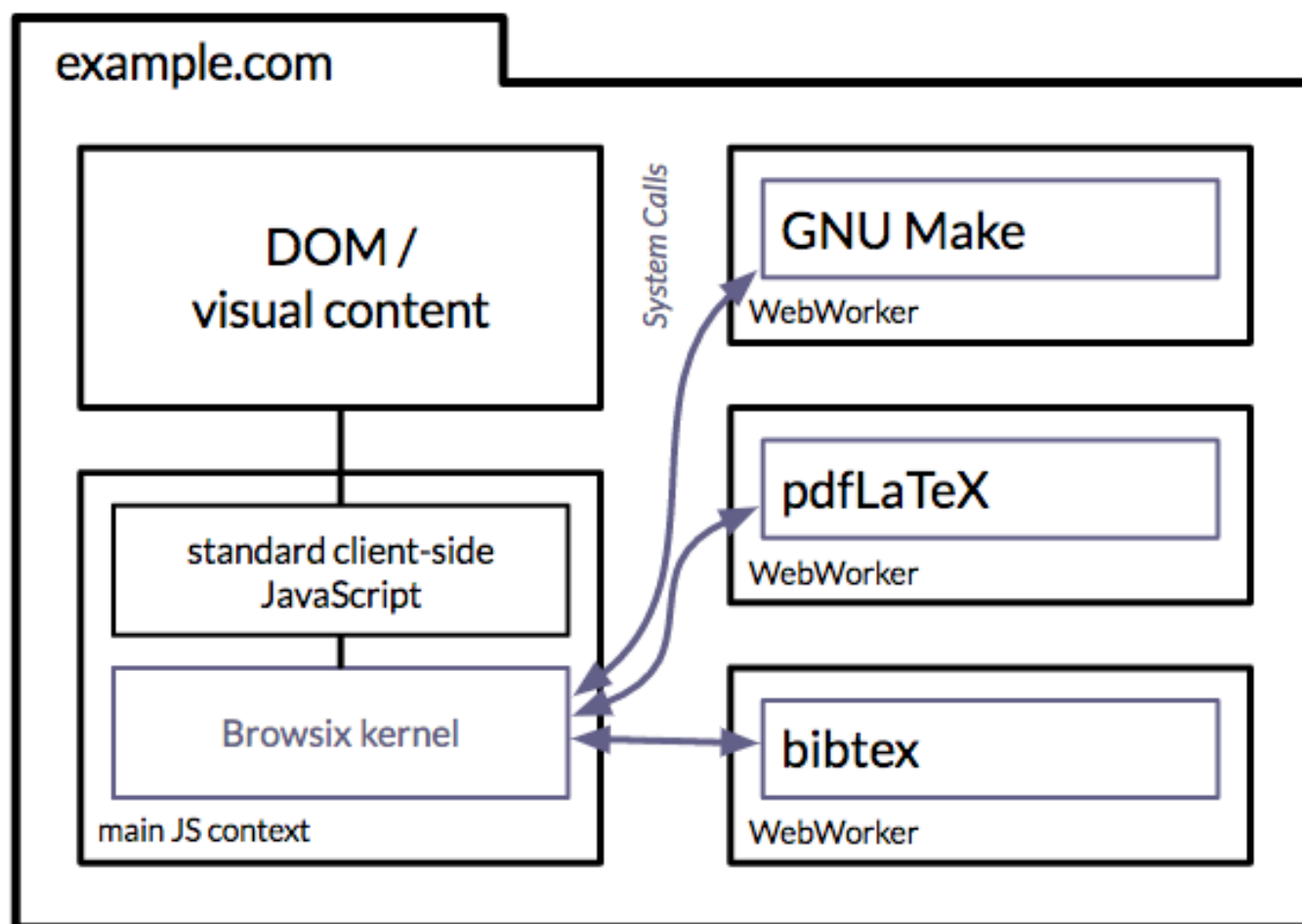
Known issues with this shell:
- 'cd' is not implemented.
- backspacing past '$' produces "interesting" results
$ echo fossbytes
fossbytes
$
```

and extends the JS runtimes for C++, C, Node.js, and Go programs. It also features a POSIX-like shell. You can try out the Browsix shell right here. It supports standard Unix utilities like ls, cat, echo, etc.

It bridges the conventional gap between your familiar operating systems and the web browser. “Browsix does this by mapping low-level Unix primitives, like processes and system calls, onto ex-

isting browser APIs, like Web Workers and postMessage,” the description on the Browsix website says.

To know more about Browsix, read the peer-reviewed paper of the same. Grab its open source code on GitHub.



<https://fossbytes.com/browsix-unix-like-operating-system-web-browser/>

BSDCan 2017 Call for Participation



BSDCan is an enormously successful grass-roots style conference. It brings together a great mix of *BSD developers and users for a nice blend of both developer-centric and user-centric presentations, food, and activities.

BSDCan 2017 will be held 9-10 June 2017, with tutorials on June 7-8, in Ottawa. They are requesting proposals for talks, and do not require academic or formal papers. If you wish to submit a formal paper, you are welcome to, but it is not required.

The talks should be written with a very strong technical content bias. Proposals of a business development or marketing nature are not appropriate for this venue.

Find out more here.

Submission Deadline: January 19, 2017

<https://www.freebsdoundation.org/news-and-events/call-for-papers/bsdcan-2017/>

Python 3.6 | Here Are The New Features

Python 3.6 release is just around the corner. This release brings many new syntax features, including formatted string literals and underscores in numeric variables. Significant improvements in CPython implementation and standard library have also been made.



Today, Python is one of the most used programming languages, and it's enjoying an extensive growth in different fields of technology. Just in case you're wishing to know more about Python's strength, go ahead and read this article.

Python 3.6 is scheduled to release on December 12, 2016. Compared to version 3.5, Python 3.6 brings many new features, additional security measures, and tries to

make things more productive and easy for developers. This article presents a short overview of the Python 3.6 features and syntax additions. Let's take a look:

Python 3.6 Features

Version 3.6 introduces some important syntax features —

Formatted string literals: These are a new kind of string literals, also called f-strings. They are prefixed with 'f' and contain certain replacement fields surrounded by {}. The replacement fields are expressions that are evaluated at runtime and formatted using format() protocol.

Underscores in numeric literals: For better readability, Python 3.6 brings the ability to use underscores in numeric literals. Now, single underscores can be used between digits and after any base specified.

Syntax for variable annotations: Here, we're talking about the standard for type annotations of function parameters. This adds the syntax for annotating different types of variables.

Asynchronous generators: Python 3.6 improves the support for native coroutines and sync / await syntax introduced in Python 3.5. Now, one can use await and yield in the same function body.

Asynchronous comprehensions: There's support for using async for in the list, set, dict comprehensions and generator expressions. Also, await expressions are now supported in all types of comprehensions.

Python 3.6 also adds new library modules, CPython implementation improvements, and many security enhancements.

You can visit Python 3.6 What's new page to learn about all the changes in detail.

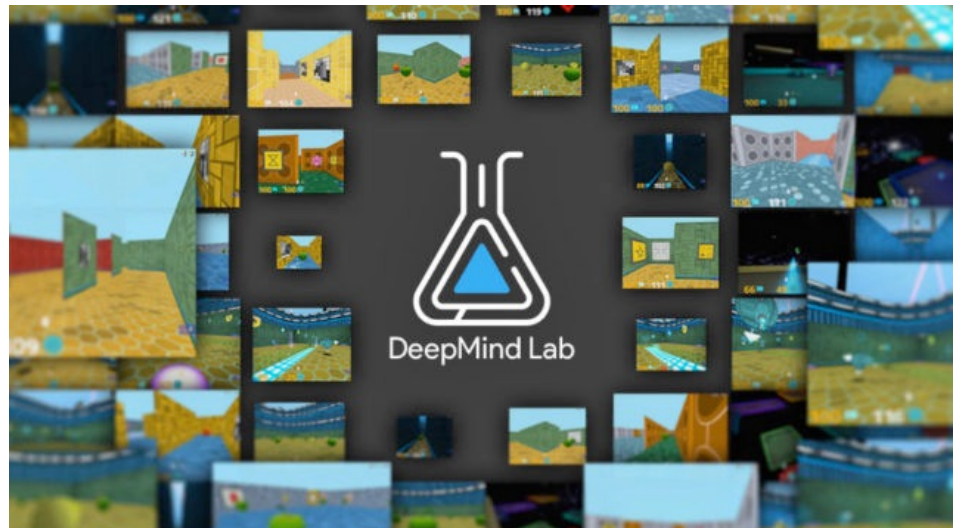
Python 3.6 Lifespan

This version will continue receiving bug fixes every 3-6 months for the next 18 months. After Python 3.7 release, a final 3.6 bug fix will be released. After that, 3.6 is expected to get security updates until December 2021.

<https://fossbytes.com/python-3-6-released-new-features/>

Google Open Sources Its AI Training Platform DeepMind Lab

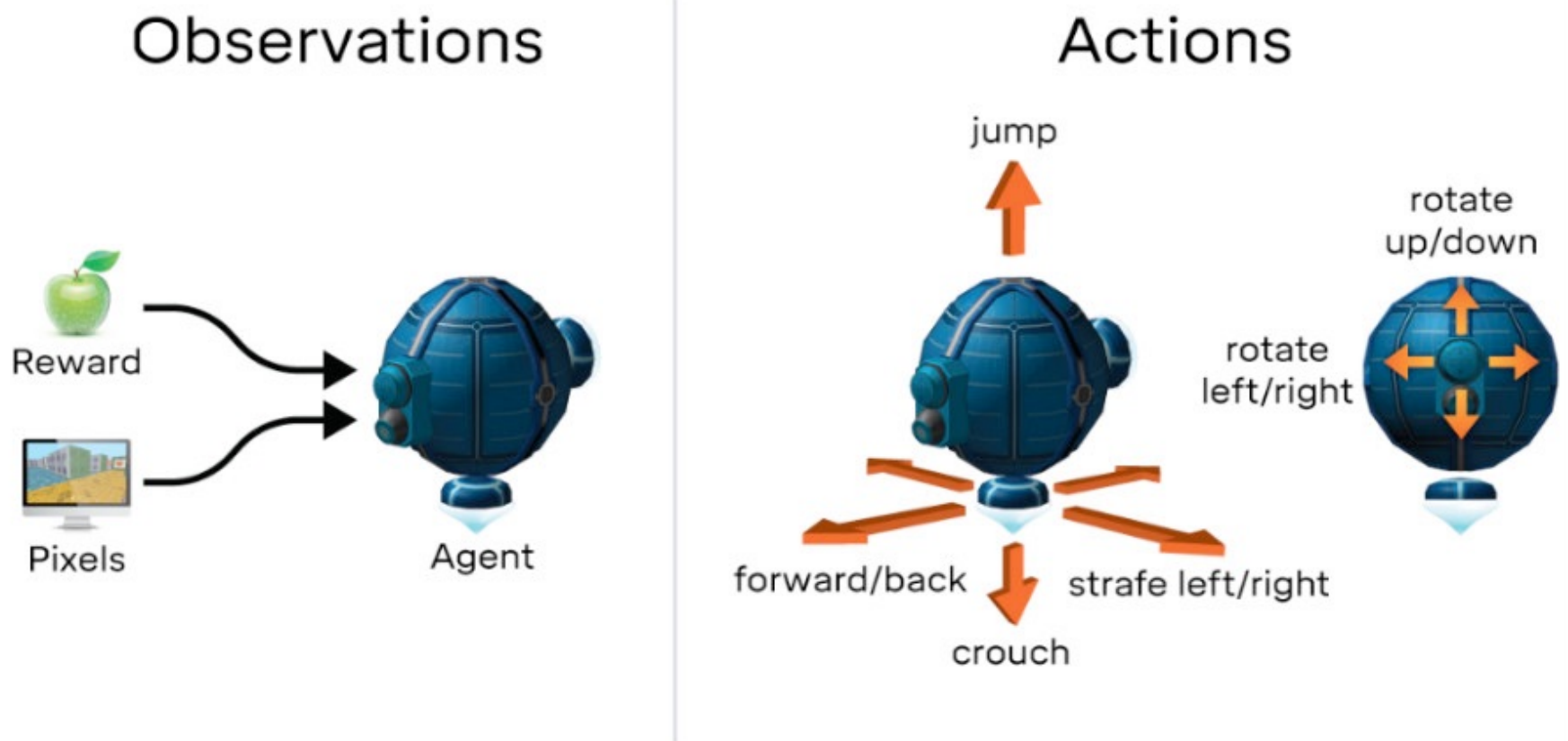
DeepMind has open sourced their AI agent training platform known as DeepMind Lab. It renders a 3D game-like environment in which the simulated agent has to float and do various tasks that enhance its cognitive abilities.



DeepMind, Alphabet's subsidiary that recently created a lip-reading AI that defeated a pro-human, has open sourced its latest project called DeepMind Lab. In a blog post, the company acknowledged the need of developing more complex training environments for AI agents and working to improve their cognitive skills.

DeepMind Lab is a platform enabling simulated environments that DeepMind says serve as “laboratories for AI research”. Basically, DeepMind Lab is a platform with 3D game-like graphics observed by a simulated agent as the first person. The agent – posed as a floating orb – navigates in the virtual 3D environment using available actions.

The agent has to perform different tasks, like moving through tough paths, collecting items, prevent itself from falling off the cliffs, etc. The purpose of this project is to focus on the agent's planning, strategy, navigation memory, first person vision, etc.



DEEPMIND LAB IS A 3D GAME-LIKE PLATFORM TO TRAIN AI AGENT.

The platform is scalable with appropriate tools available to add new levels that can be customized as per requirements. DeepMind says they have been using their platform internally and the impact it has made on their thinking about artificial intelligence is worthy of attention.

Training agents using 3D environments is vital to the improvement of their general intelligence. In fact, our rich 3D world has had a major influence on the intelligence of various living organisms, including humans. In a nutshell, the thing is, to achieve human-like intelligence, the AI has to perceive things the way they are, i.e. in 3D.

“Consider the alternative: if you or I had grown up in a world that looked like Space Invaders or Pac-Man, it doesn’t seem likely we would have achieved much general intelligence!”

<https://youtu.be/7syZ42HWhHE>

<https://fossbytes.com/google-open-sources-training-platform-deepmind-lab/>

Great Specials

On FreeBSD® & PC-BSD® Merchandise

Give us a call & ask about our
SOFTWARE BUNDLES

1.925.240.6652

\$39.95

FreeBSD 9.1 Jewel Case CD Set
or FreeBSD 9.1 DVD

\$29.95

PC-BSD 9.1 DVD

\$49.95

The PC-BSD 9.0 Users Handbook
PC-BSD 9.1 DVD

\$99.95

The FreeBSD CD or DVD Bundle

Inside each CD/DVD Bundle, you'll find:
FreeBSD Handbook, 3rd Edition
Users Guide FreeBSD Handbook, 3rd Edition, Admin Guide
FreeBSD 9.1 CD or DVD set
FreeBSD Toolkit DVD



Stylish Dress Attire
Look Your Professional Best



Comfy Apparel
Stay Warm in Zip Ups & Pullovers

T-Shirts
Lots of Styles to Choose From

FreeBSD 9.1 Jewel Case CD/DVD \$39.95

CD Set Contains:

- Disc 1** Installation Boot LiveCD (i386)
- Disc 2** Essential Packages Xorg (i386)
- Disc 3** Essential Packages, GNOME2 (i386)
- Disc 4** Essential Packages (i386)

FreeBSD 9.0 CD \$39.95

FreeBSD 9.0 DVD \$39.95

FreeBSD Subscriptions

Save time and \$\$\$ by subscribing to regular updates of FreeBSD

FreeBSD Subscription, start with CD 9.1 \$29.95

FreeBSD Subscription, start with DVD 9.1 \$29.95

FreeBSD Subscription, start with CD 9.0 \$29.95

FreeBSD Subscription, start with DVD 9.0 \$29.95

PC-BSD 9.1 DVD (Isotope Edition)

PC-BSD 9.1 DVD \$29.95

PC-BSD Subscription \$19.95

The FreeBSD Handbook

The FreeBSD Handbook, Volume 1 (User Guide) \$39.95

The FreeBSD Handbook, Volume 2 (Admin Guide) \$39.95

The FreeBSD Handbook Specials

The FreeBSD Handbook, Volume 2 (Both Volumes) \$59.95

The FreeBSD Handbook, Both Volumes & FreeBSD 9.1 \$79.95

PC-BSD 9.0 Users Handbook \$24.95

BSD Magazine \$11.99

The FreeBSD Toolkit DVD \$39.95

FreeBSD Mousepad \$10.00

FreeBSD & PCBSD Caps \$20.00

BSD Daemon Horns \$2.00



Bundle Specials!
Save \$\$\$

Just Plain Fun
Mousepads & Novelty Horns



BSD Magazine
Available Monthly



For even MORE items
visit our website today!

www.FreeBSDMall.com

SSHLock – SSH on Steroids!

by Franck Porcher, PhD

Whether locally, through an attached terminal, or remotely via SSH, standard access to computers has not evolved much in the last 40 years in spite of continuous technological advances. They still heavily rely on the traditional password-based authentication mechanism which, acting as the unique “protective door” between the world and the computer's valuable vault, most unfortunately represents a single *point of failure*.

Should this access mechanism yield to a successful password attack, the targeted system would instantly be exposed and compromised, leaving little defense in terms of security to slow the advance of the attacker towards the computer's precious content (In the case of remote logons, a single SSH layer, no matter how tightly configured, **does not** eliminate the single point of failure).

We present SSHLock, a security scheme based on SSH and devised to completely eliminate these threats in order to render computers virtually impenetrable. SSHLock is not about remote connections per se, but a more general approach to securing and strengthening the logon access mechanism to computers in order to defeat attacks on passwords and remove single point of failure access weaknesses.

SSHLock transposes into software the desirable security features found in physical security access locks to provide IT systems with a secure access mechanism in the form of a tight, confined “chamber” used as a pass-thru device between the outside world and the computer's content – two environments that should never be directly in contact with one another – and secured at each end by a reinforced “protective door” for monitoring and access control.

SSHLock brings a wealth of security features that, together, provide for a much safer access to computers over the standard password-based access mechanism, making SSHLock a safer and powerful alternative to any password-based access mechanism, including standard SSH services for remote access.

First, we provide some necessary background related to password security attacks. Then we present SSHLock, its concept and desirable security features, the principles guiding its implementation, and its overall modus operandi. The second part of the article deals with the technical details involved in manually setting-up and administering a sshlock. We conclude by presenting the SSHLock software that provides the end user with a simple interface to automate these complex tasks.

Password security attacks

The literature abounds with examples that highlight the considerable harm and damage – political, economic, and social – that may result from unauthorized access to IT systems. Discarding such omnipresent threats remains the greatest challenge of every conscientious system administrator today. Yet, access to computers still heavily relies on the weak, traditional <login , password>-based identification/authentication access mechanism.

In such a mechanism, the login is the name identifying the account one wishes to access, and is generally defined by the system administrator to reflect the final usage of the account (e.g. “webmaster”, “marcel”, “dupont”). On top of this, most IT systems still have a number of accounts whose names are publicly known and well-established by decades of standard practices and habits passed along by generations of system administrators (e.g. “root”, “www”, “apache”, “mysql”). The password is the login's counterpart : an authenticating “secret” most often left to the owner of the account to choose. Together, the clear-text login and the encrypted password constitute an access key to the account. One must know both to be granted access.

Such an access mechanism is plagued with several security weaknesses, and it not a coincidence that password security attacks remain the simplest and, therefore, the most common means to break into IT systems in order to gain unauthorized access to sensitive data and classified information :

- As the unique “security fence” between the world and the valuable content of the computer, this mechanism is a single point of failure, an undesirable feature everywhere security is paramount.
- Systems relying on such an access mechanism (Unices are no exception) are still not configured by default to use the strongest cryptography available to encrypt the passwords (hashing algorithm), resulting in insecure password databases.
- Logins and passwords are still prevalently chosen today to be both semantically significant and therefore easy to remember. Though they are perfectly valid, who wants to remember a login like “xu0i8qLKjQS” or a password like “\$, Blz! X2.718^2>e^2?;):nop @ t” ? As a consequence, and to a large extent, such access keys are generally weak from the point of view of security.
- It is still commonly believed that disclosing account names (logins), or making no effort to hide

them, does not weaken the security of this access mechanism. This is, of course, not correct, since the login is indeed an important part of the security access key. Knowing it considerably reduces the effort needed by the attacker to break the key, since only the password is now unknown. The term “password security attacks”, or simply “password attacks”, refers here to all the available means, methods and techniques that an attacker can deploy to fraudulently obtain or discover valid access keys. Password attacks are mostly used when it is not possible to take advantage of other weaknesses in an encryption system (if any exist) that would make the task easier for the attacker [1].

To fully grasp how such attacks may be carried out, one must realize that the attacker is not always that anonymous individual one may believe, lurking hidden in the shadows of a small, dark, and creepy lab located 20,000 km from home or work.

In most serious cases, the attacker knows what he is doing and does not act randomly. He may well be someone you have been in contact with, for instance:

- A personal enemy.
- An unscrupulous or vengeful staff member.
- A political opponent.
- An industrial competitor.

To obtain (steal) the credentials he needs to successfully break into your system, the attacker can rely on a number of options, including:

- **Spying**, the oldest of all methods. If the attacker is near or in the immediate vicinity of its victim, he may search out valuable hints, possibly by observing and digging around the office, under the keyboard or the desk, around the screen, inside the drawers, in the garbage bins, etc.
- **Social Engineering**, a subtle form of spying wherein the attacker tries to gather personal information directly related to the targeted account's owner, and likely to be used as potential valid credentials [2], for instance :
 - Login itself: amongst dozens of accounts, it is not that unusual that one may be using the login itself as a password.
 - Full name, date of birth, social security number, phone number,...
 - A mother may use the name or birth date of her children.
 - A man may use the name of his wife, the date of their anniversary, the name of his secretary or that of his favorite hero.

- People may use information pertaining to their other half (name, birthday date, phone number, etc.)
- The attacker may also resort to using strategies of impersonation, for instance pretending over the phone to be the system administrator, or a supervisor of higher rank, asking authoritatively for some access keys, or, conversely, he may impersonate a user and politely ask the technical support to reset the access key.
- **Dictionary attack and its variations.** In contrast to a brute-force attack where a large proportion of the access key space is searched systematically, dictionary attacks use specialized dictionaries as sources of keys deemed most likely to succeed. Dictionary attacks (for instance, against easily obtained password databases) often succeed because there is a natural tendency amongst the majority of people to choose short passwords from ordinary words and their simple variants, like appending a digit or a punctuation character, or using a zero for an O (the letter) or vice versa.

Conditions for the success of a password security attack

Whichever way the attacker may choose to mount his attack, its success relies on the two following conditions being satisfied at the same time :

- There are sufficiently weak access keys within the system so the attacker can expect to uncover some of them within a reasonable time frame (condition a).

and

- The targeted IT system allows verification, the process by which an attacker can verify the validity of guessed / generated access keys (condition b).

Counter-measure

To defeat this type of attack, the counter-measures are therefore obvious. It suffices in theory that only one of the two previous conditions is not satisfied, namely:

- (\neg “condition a”) There does not exist any weak key in the system (counter-measure a).

or

- (\neg “condition b”) The targeted IT system prevents verification (counter-measure b).

The “counter-measure a” (the negation of “condition a”) is satisfied if we can design a mandatory, non-derogatory security policy that enforces some stronger cryptography to encrypt the passwords (hashing algorithm) and requires users to use access keys virtually impossible to guess. However, the stronger the policy, the harder the keys are to remember, and the more likely the

users will find ways to not forget them, for example, by writing them down (Over the years, we have caught two thirds of our students and professional users doing exactly that!).

Therefore, we are left with no option but “counter-measure b”. However, in using the password-based authentication access mechanism, it is impossible to distinguish between an authorized user and an attacker who has acquired the access key due to the difficulty of removing “condition a”. Therefore, “counter-measure b” can only be satisfied if we remove password-based authentication access mechanism altogether or if we add a second, stronger authentication mechanism in order to remove the point of failure identified earlier.

SSHLock

SSHLock does both, and more:

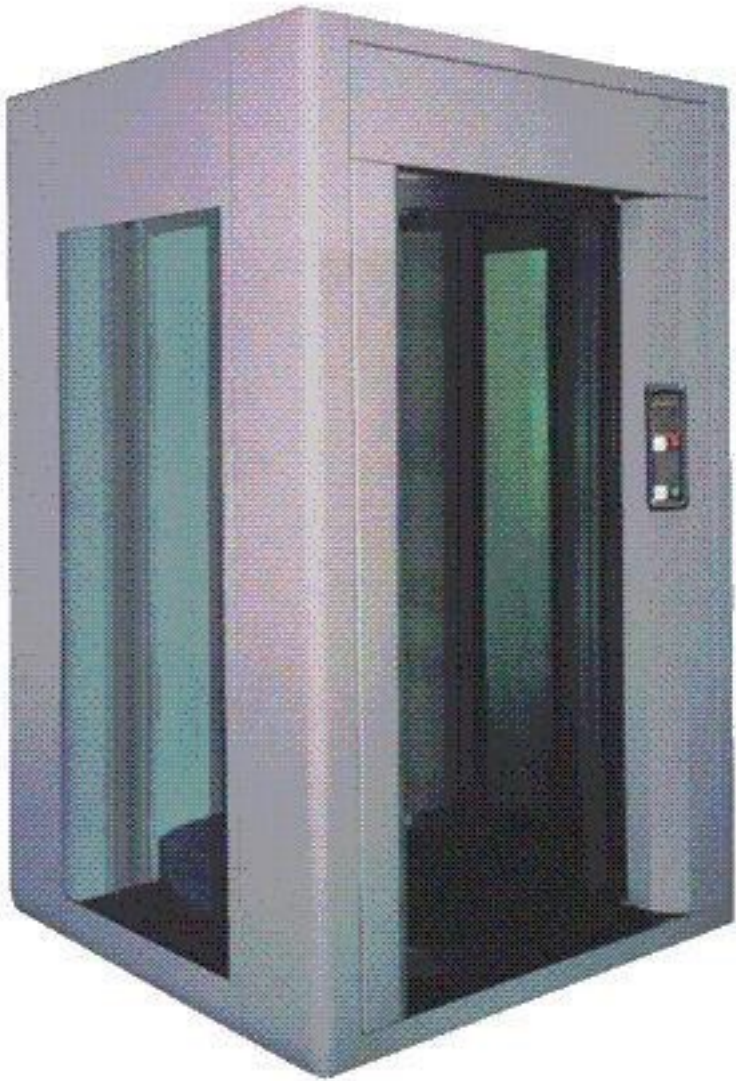
- It disables the password-based authentication access mechanism.
- It provides a sophisticated access mechanism based on two *independent* SSH layers working in series, and relying on strong public-key cryptography only, thus adding access redundancy and eliminating the single point of failure identified earlier.
- It provides a honey-pot devised to trap intruders.
- It enforces the use of strong passphrases to protect key-pairs.
- It removes the verification principle.

SSHLock draws from physical security access locks their most desirable security features, mocks them up in software, strengthens and combines them together in order to create a strong access mechanism made of a tight, confined “chamber” (the honey pot) used as a decoupling pass-thru device between the outside world and the computer's precious content. Given these two environments must not be in contact with one another, the pass-thru device is secured at each end by a reinforced “protective door” to implement monitoring and tight access control. As a result, the weaknesses identified earlier are all removed, and threats associated with password security attacks are eliminated, making breaking into an SSHLock-protected computer much harder, if not virtually impossible.

Security access lock

A lock is typically a small, secure, specialized, confined, restricted “buffer zone” (the chamber) used exclusively as a pass-thru device between two subsystems that should not directly be in contact with one another. At each extremity of the chamber is a tight, protective, reinforced door. The two doors do not open simultaneously but only one after the other.

Better known examples of this type of lock are airlocks, devices that permit the passage of people



and objects between a pressure vessel and its surroundings while minimizing the change of pressure in the vessel and loss of air from it. An airlock may be used for passage between environments of different gases rather than different pressures, to minimize or prevent the gases from mixing. [3]

A security access lock is a special type of lock used to control and monitor access to sensitive areas, for example, airport boarding areas. The most important feature of a security lock is that it is the only access to the sensitive area : any other access would only weaken the purpose of the lock. A good example of a security access lock is the type of device used by most modern banks to control and monitor access to the bank (see Figure 1).

Figure 1. Banking security device.

To enter the bank using such a security device, one must walk from the outside to the inside by passing thru the security access lock as follows:

1. You are outside the bank.
2. Arrive from the outside in front of the security access lock.
3. Wait until the security access lock becomes available.
4. Request access to enter the security access lock (usually by pushing a button placed on the outside of the security access lock's external door).
5. When access is granted, a condition usually signaled by a light turning green, push the security access lock's external door and enter the security access lock. Then allow the door to tightly close behind you.
6. You are now within the confinement of the security access lock : a small chamber between the two doors of the lock where your freedom of movement is greatly reduced due to the fact that this restricted "buffer zone" has no other destination than allowing you to pass through.

7. Once inside the security access lock and the external door has securely closed behind you, request access to the bank (usually by pushing a second button in front of the security access lock's internal door).

8. When access is granted, push the internal door of the security access lock, enter the bank, and, once again, allow the internal door to tightly close behind you.

9. Once inside the bank and the internal door has securely closed behind you, the security access lock becomes available for another customer to enter the bank.

10. You are inside the bank.

Other examples of locks include :

- Airlocks found in space vessels to allow passage between the pressurized environment inside the vessel and the unpressurized outer space outside of it.
- Airlocks found in submarines to allow passage between an air environment inside the submarine and the water environment outside of it.
- Hyperbaric chambers to allow entry and exit while maintaining the pressure difference with the surroundings.
- Cleanrooms to allow entry into protected environments in which dust, dirt particles, harmful chemicals, and other contaminants are partially excluded by maintaining the inside room at a higher pressure than that of the external world.
- Locks along ship canals connecting different seas or rivers which are not at the same altitude level (e.g. Panama's Canal, Suez Canal, Canal de Bourgogne).

A virtual security lock to access computers

SSHLock mocks-up the most desirable security features found in a physical security lock to create an sshlock, that is a virtual security access lock made of a highly restricted software environment (the confinement, or honey pot) as the single point of passage and pass-thru device between the outside world and the computer, secured by two tight access control mechanisms, one at each end, configured to work in series and allow passage only from the outside to the inside. The name sshlock is coined from implementing these access controls mechanisms using SSH.

More precisely:

1. An sshlock provides a highly restricted software environment, the confinement or honey-pot, sandwiched between two tight access mechanisms.

2. sshlock's access mechanisms work together to enforce a tight monitoring and access control at each end of the sshlock in order to prevent verification and resist intrusion.
3. The sshlock's confinement is so designed to be transparent to any authorized user (bees do not get trapped in their own honey) but to act as a honey-pot (a sticky jail) to trap an attacker/intruder. To that end, sshlock offers no facilities to figure out how to activate the sshlock's internal access mechanism to "open" the sshlock's "internal door".
4. At any time, one user at most is allowed inside the sshlock's confinement or inside the computer : when a user is inside either one, any other user is made to wait outside the sshlock until the authorized user has left both the computer and the sshlock.
5. Once a user is inside the sshlock's confinement, the external access mechanism of the sshlock becomes unavailable (its "external door" is locked).
6. Once inside the sshlock's confinement, the user must know in advance what to do to activate the internal access mechanism to open the sshlock's "internal door" and complete passing thru the sshlock. Indeed, the sshlock's restricted confinement does not allow the user to undertake anything else : one action only will work and the user must know in advance which one it is.

Peeking or poking through in search of the hidden sesame will violently eject the uninformed user outside the sshlock; such a user may never want to experience this misfortune again.

7. Once a user is inside the computer, both access mechanisms of the sshlock are unavailable (its two "doors" are locked).

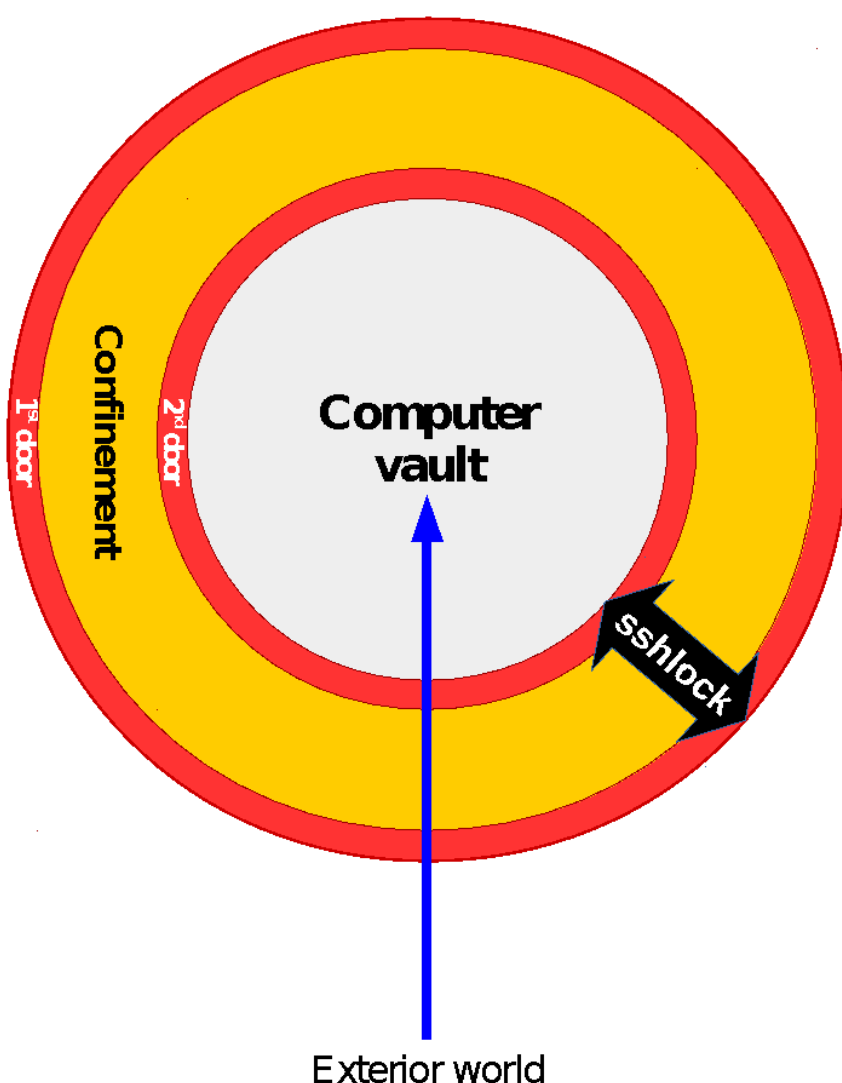


Figure 2. Access mechanisms of the sshlock.

A straightforward implementation of SSHLock can be built upon the following technologies, for their inherent strength and wide availability:

1. A hardened, highly restricted chrooted environment (jail) to mockup the physical security access lock's confinement area.
2. Two OpenSSH sshd servers serving as central access control mechanisms to mockup the security access lock's physical doors.
3. Chroot & Jail

A chroot on a Un*x operating system is an operation that changes the root of the filesystem of the process launched with the chroot(2) system call (or with the chroot(8) wrapper program). A program that runs without extra privileges in such a modified environment is guaranteed not to have access to files outside this restricted filesystem. The modified environment offered to the chrooted process and its children is commonly called a chroot jail.

Manually designing and building a tailored jail environment perfectly suited to the needs of the chrooted task is not for the faint of heart, and is mostly reserved to the well seasoned Un*x user, the main issues being:

1. To know precisely which resources to import into the jail (binaries, libraries, symbolic inks, configuration files, devices, etc.) to allow the task to run properly inside the jail.
2. To import only the strict minimum required by the task, for the more resources are imported into the jail, the weaker the jail may become.
3. To tighten ownership and access rights of the imported resources within the jail as much as possible, so the task still runs properly within the jail while not unnecessarily granting extra privilege.

Properly setting a chrooted environment always depends on the task to be run within, and can prove to be a time consuming operation due to the many steps of unavoidable trials and errors. However, patience and commitment will take one far, and we believe that the potential and overall benefits offered by the chroot mechanism far outweigh its complexities.

The chroot mechanism should not be confused with FreeBSD's jail functionality, which is a chroot on steroids aimed at providing lots of additional functionalities and more isolation than a simple chroot.

OpenSSH

OpenSSH (OpenBSD Secure Shell) is a free open-source software suite that implements the Secure Shell (SSH) cryptographic network protocol aimed at establishing secure communications

between machines connected to an IP network. It is the premier connectivity tool for remote login with the SSH protocol, encrypting all traffic to eliminate eavesdropping, connection hijacking, and other attacks. In addition, OpenSSH provides a large suite of secure tunneling capabilities, several authentication methods, and sophisticated configuration options. [4]

OpenSSH is developed by the "OpenBSD Project" (www.openssh.org) and released under the BSD license. The latest release of OpenSSH is 7.3 (2016/08/01) as of October 2016.

Reliability, robustness, strong security and simplicity-of-use have earned OpenSSH huge popularity since its inception:

- OpenSSH alone totals nearly 90%, or more, of all available SSH implementations.
- OpenSSH is freely available for many Un*x platforms (e.g. BSD, Linux, Solaris, Mac OS X, AIX, HP-UX, Cygwin).
- OpenSSH provides strong cryptography (e.g. AES, ChaCha20, RSA, ECDSA, Ed25519).
- Encryption is started before authentication: no credentials, username, password, or any other confidential information ever travels in clear (unencrypted) over the network during a SSH session.

The OpenSSH software suite provides the following tools:

- `ssh`, the OpenSSH client, a substitute for `telnet`, `rlogin`, and `rsh` to establish secure, encrypted connections to remote machines served by a SSH server:

```
$ ssh me@my.remote

$ ssh me@my.remote 'please draw me a-sheep'

$ tar cjf - ~/project | ssh me@somewhere.else 'tar xjpf - -C ~/project'
```

- `scp`, a secure substitute for `rcp`:

```
$ scp -pr ~/project me@somewhere.else:project
```

- `sftp`, a secure substitute for `ftp`:

```
$ sftp me@somewhere.else
```


- sshd, the OpenSSH server.

Default OpenSSH configuration

As stated earlier, password-based authentication access mechanism remains today's prevalent way of accessing computers worldwide. To accommodate this situation, OpenSSH is therefore most often configured by default to accept this weaker mechanism as a valid authentication procedure for remote login (See fig. 3).

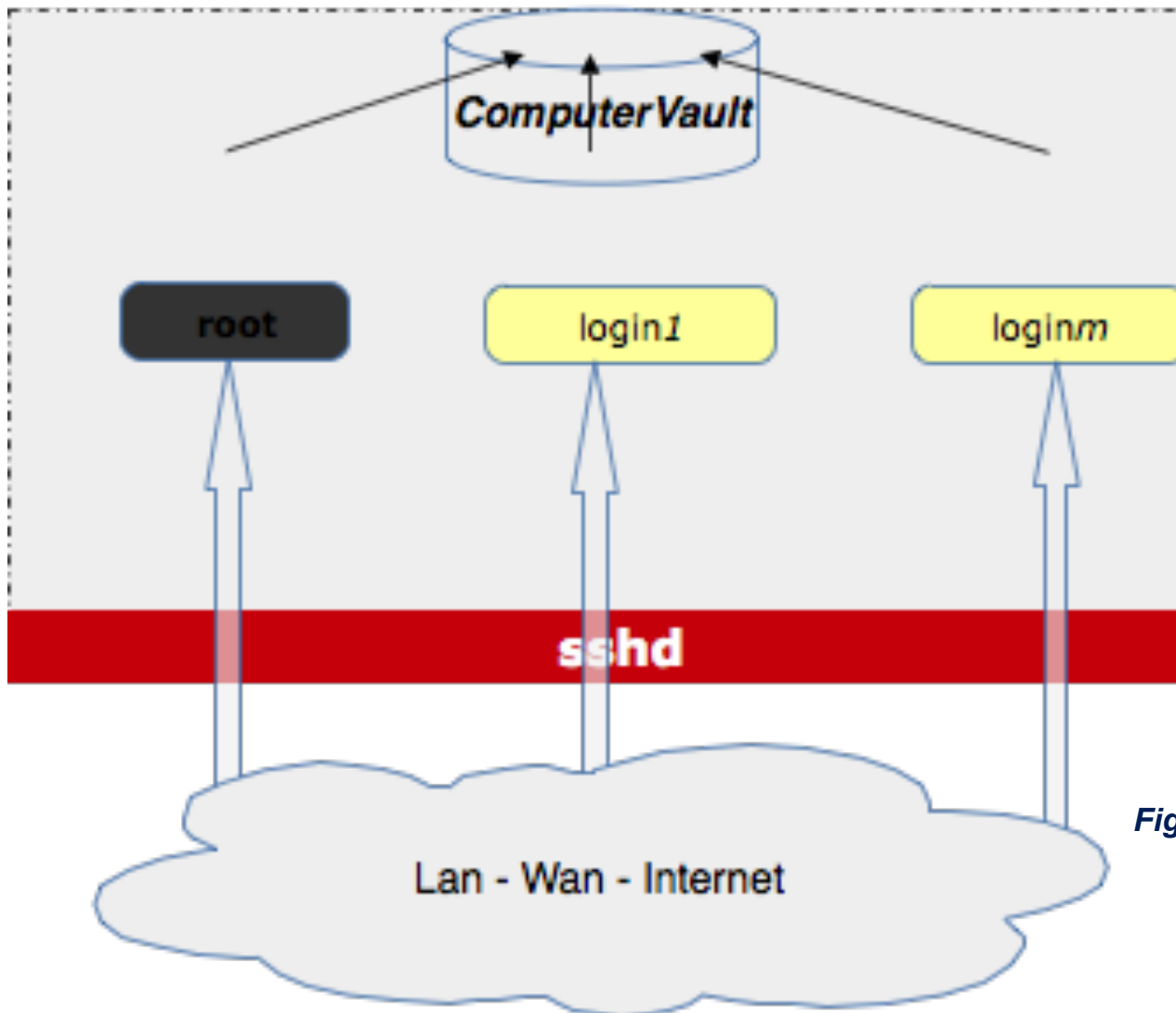


Figure 3 : SSHD default configuration.

This is most unfortunate because this standard, yet insecure, setup does not take advantage of OpenSSH's wealth of security features to remove the verification ("condition b" above) and eliminate the threat posed by password security attacks.

Single-layered access

Moreover, as a single layer access mechanism, this standard configuration of OpenSSH does not eliminate the single point of failure identified previously. The accounts of the system are now being directly exposed to the network via the sshd service, which acts as the only security fence between the outside world and the system : if an account is broken into, the whole system is compromised!

Putting SSH on steroids

Our implementation of SSHLock relies on secure configurations of chroot and OpenSSH to bring substantial isolation and security improvements over the preceding default OpenSSH configuration. This results in an overall secure, layered architecture whose sensitive access layer – previously built upon an SSH-based, single-layer access – is now made up of three layers: the sshlock. Together, these three layers create the strong and secure required access mechanism: a highly restricted software environment, the confinement (or honey-pot), sandwiched between two hardened sshd working in series to play the role of the sshlock's “protective doors”, one at each end of the confinement (see fig. 4).

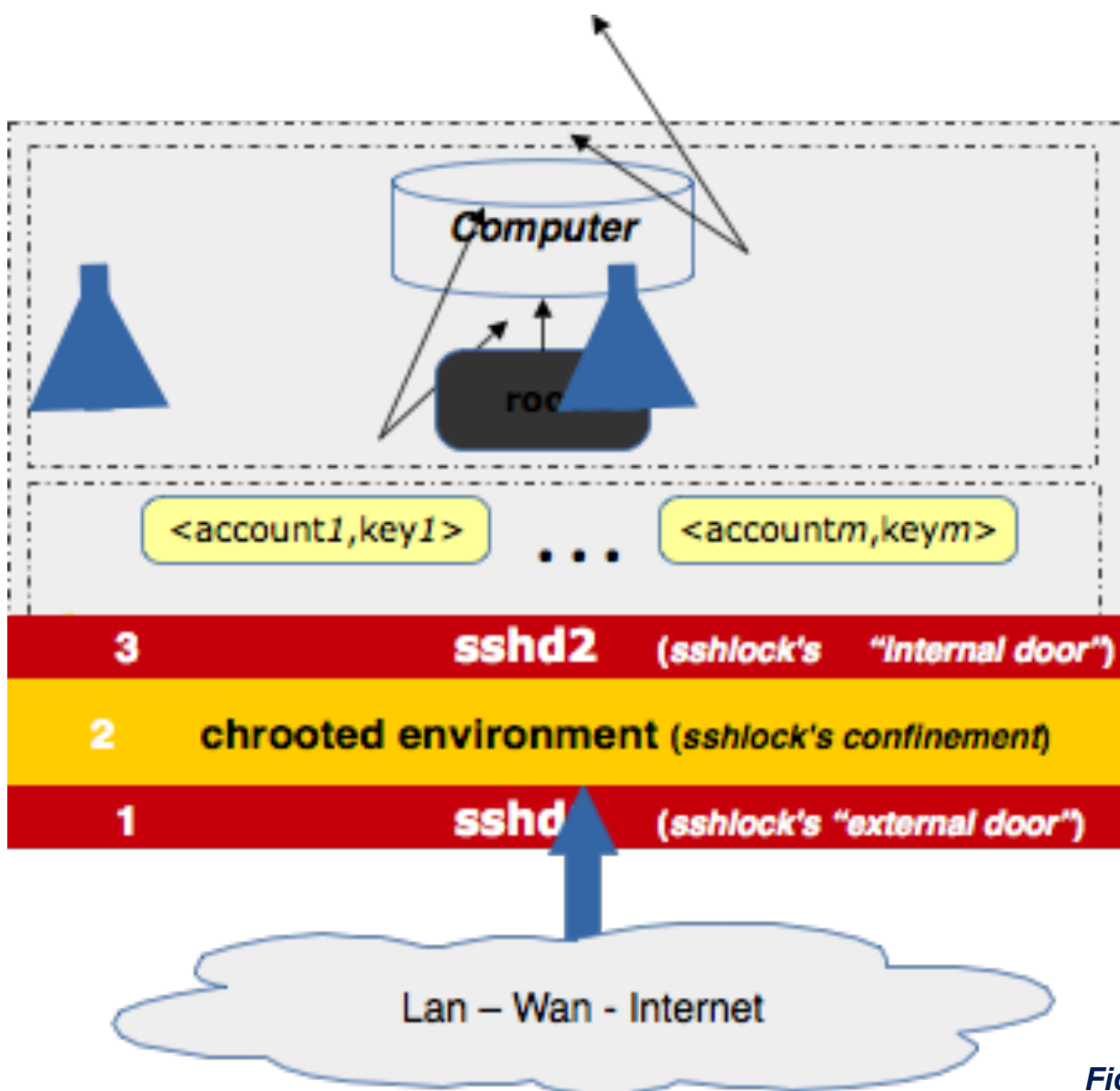


Figure 4 : SSH on steroids.

A computer protected by a sshlock as its unique access mechanism is said to be SSHLock-protected, or SSHLocked for short:

- An SSHLocked system does not allow local connections through the computer's physical terminal.
- An SSHLocked system can only be accessed from a remote computer connecting via SSH.

sshlock's main components

An sshlock is mainly comprised of the following components:

- The sshlock account, to implement the sshlock's confinement/honey-pot, configured to be the single, unique point of entry into the whole system, and therefore, the only system's account configured to accept incoming remote connections via sshd1. The sshlock account's directory and file structure are tightly set to provide the confinement, that is the highly restricted chrooted environment that serves either as a pass-thru device to authorized users from the world on the outside to the computer's vault on the inside.
- sshd1, an OpenSSH server configured to implement the external access mechanism that controls the “opening” of sshlock's “external door”. sshd1's configuration implements a restricted n-to-1 connection scheme set up to allow remote computers only (LAN, WAN, Internet) to connect to the local sshlock account only. sshd1 relies on strong public-key cryptography only for access control (password authentication is disabled). Any remote guest computer wishing to connect to the SSHLocked system must arrange to have its public key(s) installed into the sshlock account.
- sshd2, an OpenSSH server configured to implement the internal access mechanism that controls the “opening” of sshlock's “internal door”. sshd2's configuration implements a restricted 1-to-m connection scheme set up to allow the local sshlock account only to connect to (enabled) internal local accounts only. The access control to local accounts from the sshlock account relies also on strong public-key cryptography, and access to local accounts (the computer's vault) is granted on a case-by-case basis by the system administrator.

SSHLock's overall configuration of OpenSSH.

Both OpenSSH services (sshd1 and sshd2) are configured to work cooperatively in series:

- At most, one remote connection to the SSHLocked system may exist at any time (any second connection is denied until the first one is released).
- sshd2 may accept a connection only after sshd1 has accepted one (to let the user inside the sshlock's confinement).
- sshd1 and sshd2 disable password-based authentication, relying instead on the strongest public key cryptography only.

SSHLock's configuration of chroot

The sshlock's confinement (the chroot configuration) is so designed to be as transparent as possible to any authorized passing-thru user (bees do not get trapped in their own honey) but to act as

a honey-pot (a sticky jail) to trap an attacker/intruder. To that end, sshlock offers no facilities to figure out how to activate the sshlock's internal access mechanism (sshd2) to “open” the sshlock's “internal door”:

- Under the keep-it-simple principle, nothing is installed inside the chroot jail that is not absolutely necessary.
- Ownership and access rights are highly restricted, often departing from those found in standard Un*x installations.
- The sshlock account is configured to fire a no-clues, hardened restricted Bash shell (rbash) upon sshd1 accepting the incoming remote connection, so that the uninformed user (most probably an attacker/intruder) would quickly cause an error (incorrect syntax, unavailable command, violation of access rights, etc.) that will immediately result in terminating the sshd1-connection and throwing the user out of the SSHLocked system.

SSHLock's overall hardening

To fully benefit from the inherent security induced by such an improved access mechanism, no weakness should remain in the system that would allow to bypass the sshlock access mechanism. The SSHLocked system must then be hardened in the following way:

- Disable the standard password logon authentication mechanism, so the system can only be accessed remotely from remote guest computers through the SSH-based sshlock access mechanism. On most systems, this is usually done by simply locking any valid shell account, that is preventing login via password authentication without disabling the account, for instance by rendering the encrypted password into an invalid string (usually prefixing the encrypted password string with an!).

Under FreeBSD, disabling all shell accounts may be accomplished as follows:

```
1. #!/usr/bin/env bash
2.
3. ##
4. # Depending on your system, possibly check that /usr/local/bin/
   bash
5. # and /bin/bash (symlink) are included into /etc/shells
6. #
```

```
7. function list_shell_accounts () {
8.     perl -a -F: -n -E '
9.         BEGIN {%valid_shell=map{$_=>1}@ARGV;@ARGV=()} }
10.         chomp $F[6];
11.         say $F[0] if exists $valid_shell{$F[6]};
12.     ' $(cat /etc/shells | grep -E '^.+') < /etc/passwd
13. }
14.
15. for login in $(list_shell_accounts)
16. do
17.     pw lock $login
18. done
```

Under GNU/Linux systems, we will replace the statement “pw lock ...” with “passwd -l ...”

As an extra security, disable the SSHLock-system's physical console and terminals so no manual logon is ever possible from the physical terminal attached to the computer. Under FreeBSD, this is simply done by marking “off” the 5th field of any enabled entry of the file `/etc/ttys`.

Disable booting the system in single mode. Under FreeBSD, this is done by marking “insecure” the 4th field of any enabled entry of the file `/etc/ttys`, therefore forcing init to ask for the root password when the system is going to single-user mode or booted in single-user mode. Under GNU/Linux, and supposing the system is using a grub boot loader, this can be achieved by doing the following :

```
1. 1. 1. 1. # Edit the file /etc/default/grub
1. 1. 1. 2.  $ sudo vi /etc/default/grub
1. 1. 1. 3.
1. 1. 1. 4. # Add or enable the following line:
```

```
1. 1. 1. 5. GRUB_DISABLE_RECOVERY="true"

1. 1. 1. 6.

1. 1. 1. 7. # Update grub

1. 1. 1. 8. $ sudo update-grub
```

Disable booting the system from the network or an external, removable rescue media, like a CD-ROM or a USB-stick, by means of a password protected BIOS/UEFI custom setup.

SSHLock's *modus operandi*

The overall sshlock secure access mechanism operates as follows:

1. Given there is no pending or actual SSH-connection to the SSHLocked system, upon a successful, public-key-authentication-ssh-based remote connection to the SSHLocked system's sshlock account, sshd1 will grant access and start the remote connection ("opening" the sshlock's "external door") by firing the hardened restricted bash in the context of the chrooted environment, allowing the user inside the sshlock's confinement.
2. As long as the sshd1 connection lasts, sshd1 will refuse any new connection, thus blocking any other access to the system. It should also be noted that any pending connection to the SSHLocked system (a connection being authenticated) prevents any other connection to the system to take place, too.
3. Once the SSH-based access to the SSHLocked system has been granted and the user finds himself confined into the chrooted environment, very little can be done! As with the security access lock of a bank, the user must now find the way to trigger the sshlock's internal access mechanism to "open" the sshlock's "internal door" and walk into the computer's vault. This is basically the only action the user is allowed to do! However, contrary to the security access lock of a bank, there is no visible "button" to press here : the user must know what to do prior to walking-in! No peeking/poking around is allowed : the confined and highly restricted chrooted nature of the sshlock's confinement, acting as a honey-pot to the uninformed user, will silently and violently throw the user outside the system upon the first operational error:
 - Most familiar commands generally available in a normal Un*x shell session are here simply forbidden. Only three commands are enabled by default : date, scp, ssh.

- These commands cannot even be called directly but only through a command proxy whose name (“cmdproxy” by default), which can be freely chosen at SSHLock deployment time, must also be known in advance (obfuscation).
 - The slightest operational error while inside the sshlock's confinement (for example running a forbidden command, causing a syntax error, running against invalid credentials, changing directories or listing the content of directories) immediately results in violently aborting the current SSH session and connection.
4. Once inside the sshlock's confinement, and upon a successful public-key-authentication-ssh-based local connection to an internal local account (whose access has been previously granted), sshd2 will grant access and start the local connection to the chosen account, thus “opening” the sshlock's “internal door” and letting the user walk, unrestricted, into the computer.
 5. Like with sshd1, as long as the sshd2 connection lasts, sshd2 will refuse any new local connection from the sshlock account to any internal local account.

Setting up an sshlock

Setting up an sshlock can be broken into the following steps:

1. Create the local sshlock account.
2. Set up its chroot environment.
3. Set and harden the restricted shell.
4. Set sshd1.
5. Set sshd2.
6. Install public-keys, and grant accesses to internal local accounts.
7. Harden the computer (seen above).

Creating the sshlock account

Creating the sshlock account basically narrows down to creating a new user account: the sshlock account.

Given that the new entry into the system-wide `/etc/passwd` must work with respect to both referentials, namely the system-wide file system, and that of the chrooted environment, special care must be taken while setting the locations of resources associated with the home directory (6th field) and the shell (7th field) associated with the `sshlock` entry in `/etc/passwd`. Say we have the following entry:

```
sshlock:*:1001:1001:SSHLOCK:/sshlock:/restricted/rbash
```

- With respect to the system-wide filesystem, the leading slash of the absolute path `“/sshlock”` refers to the root of the system-wide filesystem, and such a path should correctly refer to the absolute location of the chrooted filesystem's root within the system-wide filesystem.
- With respect to the chrooted environment, the leading slash of the absolute path `“/sshlock”` now refers to the root of the chrooted filesystem's, that is to the resource `“/sshlock/sshlock”` within the system-wide filesystem.
- In the same manner, the shell `“/restricted/rbash”` being only referred to in the context of the chroot jail, it refers to the resource `“/sshlock/restricted/rbash”` within the system-wide filesystem.

In properly setting a chroot environment, including setting relative symbolic links within the jail, this dual view of the file-system proves to be a subtle point where most newcomers seem to stumble.

Setting up the chrooted environment

Setting up the chrooted environment requires good acquaintance with the whistles and bells of the underlying system:

- Identify the filesystem structure suitable to the desired task.
- Identify the absolute minimum resources that need to be imported inside the `sshlock` for the desired task to work reliably and consistently: which devices? Which binaries? Static or shared libraries? Which libraries (not forgetting the elf loader)? Which configuration files? Etc.
- Identify the strictest ownership and access rights that still allow the task to work reliably and consistently. This is OK, and even recommended, to depart from those found in the system-wide filesystem, often too lax.

The following listing shows an example of a possible layout for an sshlock chrooted environment (FreeBSD 10.3) :

```
1.  /home/sshlock
2.  |-- [l--x--x--x root      wheel  ]  .ssh -> ./home/sshlock/.ssh/
3.  |-- [d--x--x--x root      wheel  ]  all/
4.  |    |-- [l--x--x--x root      wheel  ]  date -> /rescue/date*
5.  |    |-- [l--x--x--x root      wheel  ]  scp -> /usr/bin/scp*
6.  |    |-- [l--x--x--x root      wheel  ]  ssh -> /usr/bin/ssh*
7.  |-- [dr-xr-xr-x root      wheel  ]  dev/
8.  |-- [d--x--x--x root      wheel  ]  etc/
9.  |    |-- [-r--r--r-- root      wheel  ]  group
10. |    |-- [-r--r--r-- root      wheel  ]  host.conf
11. |    |-- [-r--r--r-- root      wheel  ]  hosts
12. |    |-- [-r--r--r-- root      wheel  ]  inputrc
13. |    |-- [-r--r--r-- root      wheel  ]  libmap.conf
14. |    |-- [-r----- root      wheel  ]  master.passwd
15. |    |-- [-r--r--r-- root      wheel  ]  nsswitch.conf
16. |    |-- [-r--r--r-- root      wheel  ]  passwd
17. |    |-- [-r--r--r-- root      wheel  ]  profile
18. |    |-- [-r--r--r-- root      wheel  ]  protocols
19. |    |-- [-r--r--r-- root      wheel  ]  pwd.db
20. |    |-- [-r--r--r-- root      wheel  ]  services
21. |    |-- [-r----- root      wheel  ]  spwd.db
```



```
22. |    `-- [d--x--x--x root    wheel    ]  ssh/
23. |          |-- [-r--r--r-- root    wheel    ]  ssh_config
24. |          |-- [-r----- root    wheel    ]  ssh_host_ed25519_key
25. |          |-- [-r----- root    wheel    ]
    ssh_host_ed25519_key.pub
26. |          |-- [-r----- root    wheel    ]  sshd1_config
27. |          |-- [-r----- root    wheel    ]  sshd2_ac-
    counts_enabled
28. |          |-- [-r----- root    wheel    ]  sshd2_config
29. |          `-- [-r----- root    wheel    ]  sshd_banner.legal
30. |-- [d--x--x--x root    wheel    ]  granted/
31. |    |-- [l--x--x--x root    wheel    ]  cmdroxy -> /restricted/
    cmdproxy
32. |    `-- [l--x--x--x root    wheel    ]  scp -> /all/scp
33. |    |-- [d--x--x--x root    wheel    ]  home/
34. |    `-- [drwx----- sshlock  sshlock ]  sshlock/
35. |          |-- [-r--r--rw- root    wheel    ]  .bash_history
36. |          |-- [l--x--x--x root    wheel    ]  .inputrc -> /etc/
    inputrc
37. |          |-- [d--x----- sshlock  sshlock ]  .ssh/
38. |          `-- [-r----- sshlock  sshlock ]  authorized_keys
39. |-- [d--x--x--x root    wheel    ]  lib/
40. |    |-- [-r--r--r-- root    wheel    ]  libc.so.7
41. |    |-- [-r--r--r-- root    wheel    ]  libcrypt.so.5
```

```
42. | | | | -- [-r--r--r-- root      wheel    ]  libcrypto.so.7
43. | | | | -- [-r--r--r-- root      wheel    ]  libmd.so.6
44. | | | | -- [-r--r--r-- root      wheel    ]  libncurses.so.8
45. | | | | -- [-r--r--r-- root      wheel    ]  libthr.so.3
46. | | | | | | | | -- [-r--r--r-- root      wheel    ]  libutil.so.9
47. | | | | `-- [-r--r--r-- root      wheel    ]  libz.so.6
48. | | | | -- [d--x--x--x root      wheel    ]  libexec/
49. | | | | `-- [---x--x--x root      wheel    ]  ld-elf.so.1*
50. | | | | -- [dr-xr-xr-x root      wheel    ]  proc/
51. | | | | -- [d--x--x--x root      wheel    ]  rescue/
52. | | | | | | | | -- [---x--x--x root      wheel    ]  date*
53. | | | | -- [d--x--x--x root      wheel    ]  restricted/
54. | | | | | | | | -- [---x--x--x root      wheel    ]  bash*
55. | | | | | | | | -- [---x--x--x root      wheel    ]  cmdproxy*
56. | | | | | | | | `-- [l--x--x--x root      wheel    ]  rbash -> bash*
57. | | | | -- [d--x--x--x root      wheel    ]  usr/
58. | | | | | | | | -- [d--x--x--x root      wheel    ]  bin/
59. | | | | | | | | | | | | | | -- [---x--x--x root      wheel    ]  scp*
60. | | | | | | | | | | | | | | `-- [---x--x--x root      wheel    ]  ssh*
61. | | | | | | | | -- [d--x--x--x root      wheel    ]  lib/
62. | | | | | | | | | | | | | | -- [-r--r--r-- root      wheel    ]  libasn1.so.11
63. | | | | | | | | | | | | | | -- [-r--r--r-- root      wheel    ]  libcom_err.so.5
64. | | | | | | | | | | | | | | -- [-r--r--r-- root      wheel    ]  libgssapi.so.10
```

```
65. | | |-- [-r--r--r-- root wheel ] libheimbase.so.11
66. | | |-- [-r--r--r-- root wheel ] libhx509.so.11
67. | | |-- [-r--r--r-- root wheel ] libkrb5.so.11
68. | | |-- [-r--r--r-- root wheel ] libroken.so.11
69. | | |-- [-r--r--r-- root wheel ] libwind.so.11
70. | | `-- [d--x--x--x root wheel ] private/
71. | | |-- [-r--r--r-- root wheel ] libheimipcc.so.11
72. | | |-- [-r--r--r-- root wheel ] libldns.so.5
73. | | `-- [-r--r--r-- root wheel ] libssh.so.5
74. | |-- [d--x--x--x root wheel ] libexec/
75. | | `-- [l--x--x--x root wheel ] ld-elf.so.1 ->
    /libexec/ld-elf.so.1*
76. | `-- [d--x--x--x root wheel ] local/
77. | `-- [d--x--x--x root wheel ] lib/
78. | |-- [l--x--x--x root wheel ] libintl.so.8 ->
    libintl.so.8.1.5
79. | | `-- [-r--r--r-- root wheel ] libintl.so.8.1.
80. | `-- [d--x--x--x root wheel ] var/
81. | `-- [d--x--x--x root wheel ] run/
82. | |-- [-r--r--r-- root wheel ] ld-elf.so.hints
83. | `-- [-r--r--r-- root wheel ] ld-elf32.so.hints
84.
85. 23 directories, 60 files
```


sshlock's confinement (home directory)

Lines 33-38, where the user lands upon being granted access by sshd1.

1. Ownership

Apart from resources line 33, 37, 38, imposed by OpenSSH, no resource belongs to the sshlock user.

2. Access rights

• Directories

Almost all directories have the 0111 mode (traverse permission only), to prevent the sshlock user from glancing at their content. Some notable exceptions for :

- The sshlock account's home directory within the jail (line 34), which requires the owner's write permission (mode 0700) to allow resources between the outside world and the computer's vault to transit via the sshlock's confinement using scp (imagine going to the bank to replenish your account : you must be permitted to carry the cash along with you while transiting through the security lock !).

- /dev (line 7) and /proc (line 50) resources, which require the read flag (mode 0555).

- The .ssh directory of the sshlock account (line 37), which also requires the read flag (mode 0500).

• Executable binaries

All executable binaries have the mode 0111 (executable permission only), including the statically compiled elf loader, not to be forgotten (line 49).

• Binary libraries

All binary libraries have the 0444 mode (read permission only).

• Text files

Configuration files have the mode 0444 or 0400 (read permission only - lines 9-21, 23-29, 35, 36, 38).

• Exceptions

.bash_history (line 35) is the only regular file with the write bit set, to allow recording the sshlock user's activity. However, this file belonging to root, it cannot be removed nor rewritten by the

sshlock user who does not have the right to use any shell redirection mechanism, and therefore cannot invoke any command like 'date > .bash_history'.

Setting up the restricted shell

Creating a chroot jail associated to the sshlock account is only the first step into crafting a highly confined and restricted environment. The second step consists of providing the sshlock account with a restricted shell like rbash (see bash(1)) as the standard shell to be fired upon connection. This is enforced in the /etc/passwd configuration file seen in the example above.

A restricted shell is used to set up an environment more controlled than the standard shell. Restricted Bash behaves identically to Bash with the exception that the following are disallowed, forbidden or not performed, generating an error instead:

- Changing directories with the cd builtin command.
- Setting or unsetting the values of some important environment variables, like SHELL, PATH, ENV, or BASH_ENV.
- Specifying command names containing the slash character ("/).
- Specifying a filename containing a "/" as an argument to the . or source builtin commands.
- Specifying a filename containing a slash as an argument to the -p option to the hash builtin command.
- Importing function definitions from the shell environment at startup.
- Parsing the value of SHELLOPTS from the shell environment at startup.
- Redirecting output using the >, >|, <>, >&, &>, and >> redirection operators.
- Using the exec builtin command to replace the shell with another command.
- Adding or deleting builtin commands with the -f and -d options to the enable builtin command.
- Using the enable builtin command to enable disabled shell builtins.
- Specifying the -p option to the command builtin command.
- Turning off restricted mode with set +r or set +o restricted.

Hardening the restricted shell

We will harden that tight configuration even more by crafting a special rbash startup file made to enforce extra restrictions, the most important being to restrict the PATH and to exit on the first error:

```
1. #
2. # /etc/profile - rbash startup file (excerpts)
3. #
4.
5. # Restrict commands
6. export PATH=/granted
7.
8. # Exit on first command returning a non zero status (error)
9. set -e
```

Given PATH is restricted to /granted (relatively to the chroot) and the user cannot use “/” in its commands or arguments, the only available command is cmdproxy. Running any other command will result in an error that will immediately terminate the SSH session. This mandatory command will in turn launch any available commands not directly accessible (date, scp, ssh).

One may wonder whether all that seeming obfuscation is needed and/or useful. For instance, why not copy the commands under /all, instead of having links? The answer is not theoretical nor rhetorical, simply pragmatical. Never knowing what these binaries actually do and whether they may rely internally on other commands, the simplest solution is to mockup the original resource’s subtree as we install them inside the chroot jail. For instance, take the case of scp, which internally launches /usr/bin/ssh. If ssh were not to be where it is expected, that is /usr/bin/ssh, then scp would fail as a result.

As for the special directory “/restricted”, it is a particular container used to lock some necessary commands that we voluntarily do not make available to the sshlock user. For instance, though the bash shell must be available within the chroot jail to be launched upon a valid sshd1 connection, we do not want the sshlock user to have the option to launch it at whim as a subshell, which could lock him out of the restricted shell.

Setting up OpenSSH

Properly setting up OpenSSH is central to the security of SSHLock overall operation.

1. Security and cryptography

Under the “keep it simple” principle, we will disable any configuration option we will not be using, and will only resort to the strongest cryptography options.

- **Protocol:** we disable the SSH-1 protocol altogether.
- **Key exchange:** given recent revelations from ex-consultant at NSA Edward Snowden that NSA willingly inserts backdoors into software, hardware components and published standards, some researchers (including Bruce Schneier and Dan J. Bernstein) have expressed their lack of confidence in NIST-published curves, such as nistp256, nistp384, nistp521, in which it is believed that the NSA had a word to say in their definition, and for which constant parameters, including the generator point, are defined without explanation. These curves are not the most secure or fastest possible for their key sizes, and researchers think it is possible that the NSA has ways of cracking NIST curves. It is also interesting to note that SSH belongs to the list of protocols the NSA claims to be able to eavesdrop. Having a secure replacement would make passive attacks much harder if such a backdoor exists.

As such, we will be using the high-security Curve25519 Elliptic curve Diffie-Hellman algorithm proposed in 2006 by Dan J. Bernstein, its main strengths being its record-setting speeds, its constant-time run time and resistance against side-channel attacks, and its lack of nebulous hard-coded constants.

- **Key signing (server authentication):** we will be using the super fast, super secure Ed25519 Edwards curve Digital Signature Algorithm (Ed25519) public-key signature system designed by Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang, a digital scheme using a variant of the Schnorr signature based on Twisted Edwards curves, designed to be faster than existing digital signature schemes without sacrificing security.
- **Client/User authentication:** as explained in the introduction, we will disable password authentication and resort to using public-key authentication only, where we will enforce key-pairs to be protected by strong passphrases, knowing that nothing secret will ever be exchanged over the network.
- **Confidentiality:** symmetric ciphers are used to encrypt the data after the initial key exchange and authentication have been completed. We will be using the newer Bernstein's ChaCha20 symmetric cipher for Internet confidentiality, standardized in RFC 7905.

2. sshd1

sshd1, an OpenSSH server, is sshlock's external access mechanism configured to implement SSHLock's “external door”, that is for monitoring and controlling access into the sshlock's confinement from remote computers only (LAN, WAN, Internet):

- Public-key authentication only.
- Only listen to incoming remote connections; do not answer incoming local connections.
- Only accept connections to the local sshlock account; connections to any other account are prohibited.
- Upon a successful connection, lock the user into the confined, restricted environment associated to the sshlock account (hardened chroot jail).
- Do not allow more than one connection at any time, actual or pending.

```
1. #
2. # SSHD1_CONFIG EXCERPTS
3. #
4.
5. # Listen networks for incoming remote connections only
6. ListenAddress      192.168.1.12:22000
7.
8. # Disable the SSH-1 protocol altogether
9. Protocol            2
10.
11. # CLIENT/USER AUTHENTICATION
12. PubkeyAuthentication      yes
13. PasswordAuthentication    no
```

```
14.
15. # ACCESS CONTROL
16. MaxAuthTries      1
17.
18.     # Deny local connections (IP addresses of the network inter-
    faces)
19. DenyUsers          *@192.168.1.12
20. DenyUsers          *@127.0.0.1
21.
22. # Incoming Remote Connections to the local sshlock account only
23. AllowUsers          sshlock@*
24.
25. # Allow at most 1 simultaneous sshlock session at any time.
26. MaxSessions
27.
28. # Allow 1 pending unauthenticated connection at any time
29. MaxStartups        "1:100:1"
30.
31. # Lock the incoming user into the sshlock jail (OpenSSH 4.9p1 or
    later)
32. ChrootDirectory    /home/sshlock
```


3. sshd2

sshd2, an OpenSSH server, is sshlock's internal access mechanism to implement SSHLock's “internal door”, that is for controlling and monitoring access into the computer's vault via internal local accounts:

- Public-key authentication only.
- Only listen to incoming local connections; do not answer incoming remote connections.
- Only accept connections to internal accounts that have granted access to the sshlock account.
- Do not allow more than one connection at any time, actual or pending.

sshd1 and sshd2 play complementary roles and their configurations are very similar:

```
1. #
2. # SSHD2_CONFIG EXCERPTS
3. #
4.
5. ListenAddress    localhost:22000
6.
7. AllowUsers       micha@localhost
8. AllowUsers       wolfy@localhost
9. AllowUsers       baltamos@localhost
10. AllowUsers      scarlette@localhost
11. AllowUsers      bella@localhost
12. AllowUsers      sultan@localhost
13. AllowUsers      nell@localhost
14. AllowUsers      cesar@localhost
15. AllowUsers      junior@localhost
```

```
16. AllowUsers      jo@localhost
17. AllowUsers      marius@localhost
18. AllowUsers      ptitloup@localhost
19. AllowUsers      douce@localhost
20. AllowUsers      snowflower@localhost
```

4. Default ssh_config

Used by the sshlock account as a generic template to connect to any enabled internal local account via sshd2:

```
1. Host localhost
2.   Ciphers                chacha20-poly1305@openssh.com
3.   ClearAllForwardings    yes
4.   KexAlgorithms          curve25519-sha256@libssh.org
5.   MACs                   hmac-sha2-512-etm@openssh.com
6.   PasswordAuthentication no
7.   Port                   22000
8.   PreferredAuthentications publickey
9.   Protocol                2
10.  PubkeyAuthentication   yes
11.  RSAAuthentication      no
12.  SendEnv                 LANG LANGUAGE LC_*
13.  VisualHostKey          yes
```

SSHLock's management of public-keys and access to internal local accounts

Passing-thru the sshlock is done in two steps:

1. Connecting from a remote guest client to the SSHLocked-system's sshlock account via sshd1. This requires that any remote guest client wishing to connect to the SSHLocked-system has its ED25519 public key installed into the sshlock account of the SSHLocked-system.
2. Connect from the SSHLocked-system's sshlock account to the final local destination (an internal account) via sshd2. This requires that the sshlock account stores ED25519 key-pairs for each authorized user, and that the public-key counterpart of these keys are installed on a case-by-case basis into the local accounts that are granting access to the sshlock account.

- Identities

The purpose of SSHLock is to accommodate any authorized user to pass-thru the sshlock to eventually connect to an authorized local account. As such, SSHLock must allow to locally and securely store the ED25519 cryptographic key-pairs of any authorized user in order to let them access their final destination account within the SSHLocked-system via sshd2.

Identities is SSHLock's local identification mechanism by which authorized users are uniquely identified within the system, and by which their cryptographic keys are uniquely associated to them within the sshlock's confinement, to be further exported into any local account granting access.

Using the SSHLock software

Setting up an sshlock manually is not trivial; it is time consuming and error prone. Because no existing open source or commercial software met our requirements when we urgently needed a strong, reliable and perennial solution, we wrote SSHLock, the software that closely implements our SSHLock specifications. Over the years, SSHLock has become a steady product one can trust to deliver the following advantages:

- To the best of our knowledge, SSHLock is the only software of its kind that provides this secure access mechanism to computers in order to defeat password attacks and render them virtually impenetrable to attackers.
- SSHLock provides many useful options to create, obfuscate and administer an sshlock, saving weeks of anguish and headache to the savvy system administrator who only wants the job (well) done.
- SSHLock provides the end user with a simple interface that makes it a breeze to remotely connect to SSHLocked-system.

1. Supported platforms

In its current version (1.21 as of November 2016), SSHLock primarily targets FreeBSD systems (FreeBSD, PC-BSD, FreeNAS, ...) and Debian GNU/Linux software distributions (Ubuntu family, Mint, Knoppix). However, people are welcome and encouraged to request that we port this software to other platforms.

2. Where to get the SSHLock software

SSHLock can be freely downloaded by visiting the page:

<http://www.franckys.com/products/sshlock/index.html>

3. Software requirement

SSHLock will work best on FreeBSD and GNU/Linux platforms fairly up to date. SSHLock will not work with versions of OpenSSH prior to 4.9p1 (we need the ChrootDirectory option), and versions of Bash prior to 4.0 (we use some Bash advanced features :().

4. Obfuscation

SSHLock provides safe defaults out of the box for a number of parameters:

Parameter	Option	Default value
Name of the sshlock account	-u	sshlock
root of the chroot filesystem	-r	/home/sshlock*
Non standard SSH Port	-p	22000
Name of the Command-proxy	-w	cmdproxy

(*) Set by the system.

However, customizing these parameters on a case-by-case basis is highly recommended to make your own SSHLock deployment less predictable and therefore more secure.

4. SSHLock main operations

Using SSHLock is best explained by walking through a real session. Please be aware that SSHLock requires root privileges, and that to stay on the safe side, it is recommended to keep a console open at all times on the host system while initially installing and running the software.

Here is a typical scenario: say Franck is installing SSHLock on his server at work (freebsd.at.work – in blue below) so he can safely administer it remotely from home (freebsd.at.home – in dark yellow below).

```
#

# freebsd.at.home (remote guest client)

#

# List the SSHLocked-systems I am already administering from home

[franck@freebsd.at.home: ~]$ cd ~/sshlock; ls

ai.pf/    franckys.com/    ipbx.sv.ca/    webserv.sv.ca/

# Create a new entry for my freebsd server at work :

[franck@freebsd.at.home: ~/sshlock]$ mkdir freebsd.at.work; cd
freebsd.at.work

#

# freebsd.at.work (sshlock service)

#
```

```
# Check my SSHLock kit

[franck@freebsd.at.work: ~]$ cd ~sshlock ; ls

sshlock      cmdproxy

# Create an sshlock with "name:mysshlock", "chrootfs:/mysshlock",
"port:22022",

# "cmdproxy:please", "machine name:freebsd.at.work"

[franck@freebsd.at.work: ~/sshlock]$ sudo ./sshlock -u mysshlock -r /mys-
sshlock -p 22022 -w please -N freebsd.at.work -C

==

== sshlock - Version 1.21 -- lundi 24 octobre 2016, 04:55:09
(UTC+0100) ==

==

== System: freebsd

== Root:    root

== Wheel:   wheel

== Checking environment ==

...

== Creating sshlock account ==

...

>>> -----

>>> Please find "sshlock-keygen", a script to generate ED25519 crypto-
graphic

>>> key-pairs you need on your guest client machines to connect to
any
```

```
>>> SSHLocked-system.  
  
>>>  
  
>>> Please find "sshlock-connect-freebsd.at.work", a script to connect to  
connect to  
  
>>> "freebsd.at.work" (public DNS name) from any guest client machine.  
chine.  
  
>>>  
  
>>> Please export both scripts to any machine you wish to use as  
guest clients  
  
>>> to connect to the SSHLocked-system at "freebsd.at.work", and learn  
how to use  
  
>>> them:  
  
>>>  
  
>>> 1) Generation of ED25519 cryptographic key-pairs on your guest  
client  
  
>>> machine  
  
>>> $ sshlock-keygen -h  
  
>>> 2) Export and install your keys on freebsd.at.work (see helps)  
  
>>> 3) Connection to freebsd.at.work :  
  
>>> $ sshlock-connect-freebsd.at.work -h
```



```
# Display the resulting sshlock current configuration (stored by default in file

# /etc/sshlock.conf)

[franck@freebsd.at.work: ~/sshlock]$ sudo cat /etc/sshlock.conf

CONFIG_SSHLOCK_DNSNAME="freebsd.at.work"

CONFIG_SSHLOCK_CONFIGFILE="/etc/sshlock.conf"

CONFIG_SSHLOCK_USERNAME="mysshlock"

CONFIG_SSHLOCK_HOMEDIR="/mysshlock"

CONFIG_SSHLOCK_SSHD_PORT="22022"

CONFIG_SSHLOCK_PROXY_WRAPPER="please"

CONFIG_SSHLOCK_CLIENT_SSHCONFIG="sshlock-connect"

CONFIG_SSHLOCK_ACTION='help'


# Follow the instructions above and forward the 2 scripts to my guest machine

# freebsd.at.home

[franck@freebsd.at.work: ~/sshlock]$ ls

sshlock sshlock-keygen cmdproxy      sshlock-connect-freebsd.at.work


[franck@freebsd.at.work: ~/sshlock]$ scp sshlock-keygen
sshlock-connect-freebsd.at.work
franck@freebsd.at.home:sshlock/freebsd.at.work

Password for franck@freebsd.at.home:

sshlock-keygen                100% 2846   2.8KB/s   00:00 ETA

sshlock-connect-freebsd.at.work 100% 3653   3.6KB/s   00:00 ETA
```

```
#

#  freebsd.at.home  (client)

#

# Check that I received the 2 scripts from franck@freebsd.at.work

[franck@freebsd.at.home: ~/sshlock/freebsd.at.work]$ ls -l

total 12

-r-xr-xr-x  1 franck  franck  3653 Nov 21 20:41
sshlock-connect-freebsd.at.work

-r-xr-xr-x  1 franck  franck  2846 Nov 21 20:41 sshlock-keygen

# Create an ed25519 key-pair that I will use to connect to my
sshlocked-system

# at freebsd.at.work

[franck@freebsd.at.home: ~/sshlock/freebsd.at.work]$ ./sshlock-keygen
-i franck@freebsd.at.home -P 'Dumb!Pa55phrase'

>>>

>>> Your new key pair is now available at :

>>>

>>>    ~/.ssh/ed25519.franck@freebsd.at.home    (private key)

>>>    ~/.ssh/ed25519.franck@freebsd.at.home.pub    (public key)

>>>

>>> You now need to import this public key to your remote SSHLocked-
system
```

```
>>>
```

```
>>> 1) Forward the file: [ ~/.ssh/ed25519.franck@freebsd.at.home.pub ] to  
the
```

```
>>> sysadmin of the sshlocked-system you wish to connect into,  
along with
```

```
>>> the dentity you wish to use.
```

```
>>>
```

```
>>> 2) The sysadmin of the remote SSHLocked-system will run the fol-  
lowing command
```

```
>>> to install your key and grant you access into the system :
```

```
>>> $ sshlock -i "franck@freebsd.at.home" -K  
ed25519.franck@freebsd.at.home.pub
```

```
>>>
```

```
>>> When this is done, you will be able to connect to the remote  
SSHLocked-system
```

```
>>> using the command :
```

```
>>> $ sshlock-connect-freebsd.at.work franck@freebsd.at.home
```

```
>>>
```

```
# Let's check my new ed25519 key-pair
```

```
[franck@freebsd.at.home: ~/sshlock/freebsd.at.work]$ ls
```

```
authorized_keys
```

```
ed25519.franck@ai.pf
```

```
ed25519.franck@ai.pf.pub
```

```
ed25519.franck@franckys.com
```

```
ed25519.franck@franckys.com.pub
```

```
ed25519.franck@freebsd.at.home
```

```
known_hosts
```

```
# Follow the instruction: export my new public key to my sshlocked-  
system at work
```

```
[franck@freebsd.at.home: ~/sshlock/freebsd.at.work]$ scp
```

```
~/ .ssh/ed25519.franck@freebsd.at.home.pub franck@freebsd.at.work:sshlock
```

```
Password for franck@freebsd.at.work:
```

```
ed25519.franck@freebsd.at.home.pub          100%   101 0.1KB/s   00:00 ETA
```

```
#
```

```
# freebsd.at.work  (sshlock service)
```

```
#
```

```
# Check that I received the franck@freebsd.at.home's public key
```

```
[franck@freebsd.at.work: ~/sshlock]$ ls
```

```
ed25519.franck@freebsd.at.home.pub  
sshlock-connect-freebsd.at.work
```

```
sshlock          sshlock-keygen      cmdproxy
```

```
# Install that public key under the identity 'franck@freebsd.at.home'
```

```
[franck@freebsd.at.work: ~/sshlock]$ sudo ./sshlock -i
```

```
franck@freebsd.at.home -K ed25519.franck@freebsd.at.home.pub
```

```
> Imported public-key: [ed25519.franck@freebsd.at.home.pub] for identity:  
[franck@freebsd.at.home] successfully installed!
```



```
> Imported public-key: [ed25519.franck@freebsd.at.home.pub] for identity:
[franck@freebsd.at.home] successfully installed!

1 imported public-keys were successfully installed in sshlock

# Grant identity 'franck@freebsd.at.home' access to my local account
'franck'

[franck@freebsd.at.work: ~/sshlock]$ sudo ./sshlock -i
franck@freebsd.at.home -k franck -P 'An0ther!Dumb&Passphra53'

Generating an ed25519 key pair for identity: [franck@freebsd.at.home]

> Access to local account: [franck] was successfully granted to iden-
tity: [franck@freebsd.at.home].

1 local accounts/identityies were enabled

Configuring/reconfiguring OpenSSH sshd2 server

# Review allgranted access rights

[franck@freebsd.at.work: ~/sshlock]$ sudo ./sshlock -l

== Identities authorized to connect to this computer via sshlock ==

franck@freebsd.at.home: ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIKMh8VdDI50WIvEOmkeWNuu0yxFH+qh5LcU/D9PKxkYW
franck@freebsd.at.home

1 accesses are currently granted to this computer via sshlock

== Accounts accepting local connections from the sshlock ==

franck enabled for: franck@freebsd.at.home

1 accesses are currently granted to local accounts
```

```
# Everything seems ok : start the sshlock service

[franck@freebsd.at.work: ~/sshlock]$ sudo ./sshlock -S

== Enabling sshlock ==

Mounting volume: [/mysshlock/dev]

Mounting volume: [/mysshlock/proc]

Stopping sshd daemons: [674]

SSHLock service is running: [7972 7974]


#

#  freebsd.at.home (client)

#


# Let's manually connect to my new SSHLocked-system freebsd.at.work

[franck@freebsd.at.home: ~/sshlock/freebsd.at.work]$
./sshlock-connect-freebsd.at.work -i franck@freebsd.at.home
```

Legal notice

Any fraudulent or non authorized connexion to this
computer equipment by any means will be prosecuted
according to the law.

Information légale

L'auteur de connexions frauduleuses ou non autorisées
à cet équipement informatique, par quel que moyen que
ce soit, sera poursuivi judiciairement selon la loi
en vigueur au moment de l'infraction.

```
Enter passphrase for key  
' /home/franck/.ssh/ed25519.franck@freebsd.at.home':
```

```
Last login: Mon Nov 21 22:28:25 2016 from 192.168.1.1
```

```
# I am in. It works! I just passed the sshlock's "external door" and  
I'm now
```

```
# within the sshlock's very restricted confinement.
```

```
(session monitored) 06:32:08 mysshlock@freebsd.at.work [~] $
```

```
# Be cautious man... see if I can make it thru to my final destina-  
tion !
```

```
(session monitored) 06:32:08 mysshlock@freebsd.at.work [~] $ please ssh  
-i ~/.ssh/franck@freebsd.at.home franck@localhost
```

```
The authenticity of host '[localhost]:22022 ([127.0.0.1]:22022) '  
can't be established.
```

```
ED25519 key fingerprint is  
SHA256:rchWOsE+97/JADu6sTh+bqTORdbj5o1MVv3dO8OUvH0.
```

```
+-- [ED25519 256] --+
|
|
|
|
| . . o |
| = S o . . |
| =.B = . . = |
| o& B . . + +|
| .oBo@ + o . =E|
| o==B.o ..=. .+|
+---- [SHA256] ----+
```

No matching host key fingerprint found in DNS.

Are you sure you want to continue connecting (yes/no)? Yes

Legal notice

Any fraudulent or non authorized connexion to this
computer equipment by any means will be prosecuted
according to the law.

Information légale

L'auteur de connexions frauduleuses ou non autorisées

L'auteur de connexions frauduleuses ou non autorisées à cet équipement informatique, par quel que moyen que ce soit, sera poursuivi judiciairement selon la loi en vigueur au moment de l'infraction.

```
Enter passphrase for key '/mysshlock/.ssh/franck@freebsd.at.home':
```

```
Last login: Mon Nov 21 19:45:17 2016 from 192.168.1.12
```

To see the last 10 lines of a long file, use "tail filename". To see the

first 10 lines, use "head filename".

```
-- Dru <genesis@istar.ca>
```

```
# It worked! I'm now remotely connected to franck@freebsd.at.work from home !
```

```
[franck@freebsd.at.work: ~]$ cd sshlock; sudo ./sshlock -s
```

```
SSHLock service is running: [7972 7974]
```

```
# I can now administer my new sshlock system from home, let's harden it :)
```

```
[franck@freebsd.at.work: ~/sshlock]$ sudo ./sshlock -FH
```

```
== Hardening the sshlock ==
```

```
> Locking accounts to prevent console login :
```

```
Account: [root] now locked
```

```
Account: [franck] now locked
```

```
Account: [wolfy] now locked
```

```
> Making booting in single mode require root password
```

```
# Let's add some command to the sshlock's confinement (not recommended) :
```

```
[franck@freebsd.at.work: ~/sshlock]$ sudo ./sshlock -B whoami,ls
```

```
== Adding resource : binary ==
```

```
Binary: [whoami]
```

```
Static version: [/rescue/whoami]
```

```
Binary's final source: [/rescue/whoami]
```

```
Destination: [/mysshlock//rescue/whoami]
```

```
== Adding resource : binary ==
```

```
Binary: [ls]
```

```
Static version: [/rescue/ls]
```

```
Binary's final source: [/rescue/ls]
```

```
Destination: [/mysshlock//rescue/ls]
```

```
# Leave the vault and go back to the sshlock's confinement...
```

```
[franck@freebsd.at.work: ~/sshlock]$ exit
```

```
logout
```

```
Connection to localhost closed.
```

```
# I'm back to the sshlock's confinement, with 2 new commands at my
fingertip ;)

(session monitored) 06:43:11 mysshlock@freebsd.at.work [~] $ please
whoami

mysshlock


# (I don't have the right to list the content of my ~/.ssh)

# Let's generate an error and witness first hand the violent eject !

(session monitored) 06:43:24 mysshlock@freebsd.at.work [~] $ please
ls .ssh

ls: .ssh: Permission denied

Connection to freebsd.at.work closed.


# Back home, it all worked, let's have a cold beer :)

[franck@freebsd.at.home: ~/sshlock/freebsd.at.work]$


#

# freebsd.at.work (sshlock service)

#


# Everything works. Let's deploy sshlock as an autostart service

[franck@freebsd.at.work: ~/sshlock]$ sudo ./sshlock -I

Autostart service 'sshlock' successfully installed!
```

```
# Let's see...
```

```
[franck@freebsd.at.work: ~/sshlock]$ sudo /etc/rc.d/sshlock status
```

```
SSHLock service is running: [7972 7974]
```

Word of caution

1. Creating multiple sshlocks within the same machine

SSHLock does preclude creating multiple sshlocks within a given machine. However, it should be mentioned that doing so does not increase the security. In such a situation, special care should be taken to ensure the following:

- The multiple sshlocks should not compete for the same ports (option -p).
- The multiple sshlocks should use different sshlock names (option -u) and different roots (the best is to leave that blank and let the system use safe defaults).
- The configuration files do not compete for the same location (option -f).
- Only one can be installed by default as an autostart service (they compete for the same system files).

2. Adding more commands to the base

Though adding more commands to the chrooted environment is possible through options -B and -b, this is never necessary since the only intended purpose of the sshlock is to pass thru the sshlock's confinement.

3. Not using passphrase to protect your ed25519 keys

SSHLock only runs on the machine that is to be protected, not on the remote clients. Therefore, even though the “sshlock-connect-<host>” scripts do their best, SSHLock cannot enforce the remote user to use strong passphrases to protect the generated ed25519 key-pairs.

Not using strong passphrases is strongly discouraged, as anyone getting your keys will be able to impersonate you and successfully connect to the first stage of the SSHLocked-system. Though this person will likely not go too far within the sshlock's confinement, you will still have to prove that it was not you!

PASS-THRU mode

The command “sshlock-connect-<host>” generated by SSHLock during the creation of an sshlock on a host system, and to be exported to all remote guest clients wishing to connect to this SSHLocked host, can be used in several ways. One of them is the pass-thru mode, which allows the user to connect directly to his final local account on the SSHLocked-system without dealing with all the complexities involved when passing manually through the sshlock, including the highly restricted sshlock's confinement.

For instance, say that we are using the identity [franck@freebsd.at.home](#) (as seen above during our little scenario) to connect directly to the account franck on the SSHLocked-system freebsd.at.work using this script. It suffices to enter the following command, followed by the two required passphrases in sequence, and we are all done:

```
#

#  freebsd.at.home  (client)

#

# List the SSHLocked-systems I am already administering from home

[franck@freebsd.at.home: ~]$ cd ~/sshlock; ls

ai.pf/      franckys.com/    ipbx.sv.ca/     webserv.sv.ca/

# Create a new entry for my freebsd server at work :

[franck@freebsd.at.home: ~/sshlock]$ mkdir freebsd.at.work; cd
freebsd.at.work

# Notice the addition of the option -p (pass-thru)

[franck@freebsd.at.home: ~/sshlock/freebsd.at.work]$

./sshlock-connect-freebsd.at.work -i franck@freebsd.at.home -p
```

Legal notice

Any fraudulent or non authorized connexion to this computer equipment by any means will be prosecuted according to the law.

Information légale

L'auteur de connexions frauduleuses ou non autorisées à cet équipement informatique, par quel que moyen que ce soit, sera poursuivi judiciairement selon la loi en vigueur au moment de l'infraction.

Enter passphrase for key

' /home/franck/.ssh/ed25519.franck@freebsd.at.home':

bash: warning: setlocale: LC_ALL: cannot change locale (fr_FR.UTF-8):
No such file or directory

The authenticity of host '[localhost]:22022 ([127.0.0.1]:22022) '
can't be established.

ED25519 key fingerprint is

SHA256:pj+7LrAGfTN08DNZa5vgEWO/kZZV+p+Tgpg7hRTmJRg.

```
|      Eo      ..  |
|      ..++....  |
|      +o*+=.    |
|      .  OoO   .  |
|      . .  oSB.=  .  |
|      .  o +o..*..  .o|
|      . +.o o..  . +.|
|      o  ....    .  .|
|      .    o==.    |
```

+----[SHA256]-----+

No matching host key fingerprint found in DNS.

Are you sure you want to continue connecting (yes/no)? Yes

Failed to add the host to the list of known hosts
(/home/sshlock/.ssh/known_hosts).

Legal notice

Any fraudulent or non authorized connexion to this
computer equipment by any means will be prosecuted
according to the law.

Information légale

L'auteur de connexions frauduleuses ou non autorisées

à cet équipement informatique, par quel que moyen que ce soit, sera poursuivi judiciairement selon la loi en vigueur au moment de l'infraction.

```
Enter passphrase for key '/home/sshlock/.ssh/franck@freebsd.at.home':
```

```
Last login: Wed Nov 23 05:02:15 2016 from 127.0.0.1
```

```
The Moon is Waning Crescent (29% of Full)
```

```
# I have reached franck@freebsd.at.work, my local, non restricted account at
```

```
# work, after successfully passing automatically thru the sshlock.
```

```
05:20:38:152:0 [franck@freebsd.at.work: ~ ]$ ls
```

```
projets/      system/
```

```
freebsd.at.work
```

We can also launch commands directly:

```
#  
  
# freebsd.at.home (client)  
  
#  
  
# Let's automatically connect to my final destination on my new  
sshlocked  
  
# system freebsd.at.work
```

```
# Notice the addition of the option -p (pass-thru)

[franck@freebsd.at.home: ~/sshlock/freebsd.at.work]$
./sshlock-connect-freebsd.at.work -i franck@freebsd.at.home -p ls

...

Enter passphrase for key
'/home/franck/.ssh/ed25519.franck@linux.at.home':

...

Enter passphrase for key '/home/sshlock/.ssh/franck@linux.at.home':

projets

system

Connection to freebsd.at.work closed.

# List the SSHLocked-systems I am already administering from home
[franck@freebsd.at.home: ~]$ cd ~/sshlock; ls

ai.pf/      franckys.com/  ipbx.sv.ca/   webserv.sv.ca/

# Create a new entry for my freebsd server at work :

[franck@freebsd.at.home: ~/sshlock]$ mkdir freebsd.at.work; cd
freebsd.at.work
```


SSHLock's pass-thru mode perfectly illustrates the elegance of SSHLock's interface, as simple to use as that of standard SSH, and that makes it a breeze to remotely connect to and use an SSHLocked-system:

```
[SSH]    $ ssh -i franck@freebsd.at.home franck@freebsd.at.work
```

```
[SSHLock] $ sshlock-connect-freebsd.at.work -i franck@freebsd.at.home
```

```
[SSH]    $ ssh -i franck@freebsd.at.home franck@freebsd.at.work ls
```

```
[SSHLock] $ sshlock-connect-freebsd.at.work -i franck@freebsd.at.home -p ls
```

SSHLock versus SSH

SSHLock is not about remote connection per se, but a more general approach to securing and strengthening the logon access mechanism to computers in order to defeat attacks on passwords and render computers virtually impenetrable.

No matter how tightly configured, a single-layer SSH for remote access – by far the standard OpenSSH Configuration – is a simple line of defense that does not eliminate the single point of failure identified earlier. In other words, a single-layer SSH does not isolate the computer from the outside world : if it yields to an attacker, the computer is compromised.

In comparison, by providing a tightened passing-thru device (the sshlock's confinement, aka honey pot), SSHLock isolates the computer's valuable content from the outside world : as opposed to the previous scenario, should the sshlock's first SSH layer yield to an attacker, the intruder is trapped by the honey pot instead of walking free inside the computer, without any possibility to peek and poke around!

SSHLock is not about having two cryptographic keys, but about having two independent SSH-based access layers working cooperatively in series, and tightly configured to let an authorized user pass-thru the honey pot in two mandatory steps.

The security benefits brought in by SSHLock in its own attempt to remedy the weaknesses identified earlier cannot be achieved by means of a simpler set-up. Should one set oneself to remove one of the two access layers or the sshlock's confinement, or both, it won't be SSHLock any more!

The security benefits brought in by SSHLock in its own attempt to remedy the weaknesses identified earlier cannot be achieved by means of a simpler set-up. Should one set oneself to remove one of the two access layers or the sshlock's confinement, or both, it won't be SSHLock any more!

Conclusion

We have presented SSHLock, a security scheme based on SSH to eradicate the threats commonly associated with the traditional password-based standard access mechanism to computers.

SSHLock has been carefully devised to transpose into software the desirable security features found in physical security access locks in order to provide IT systems with the means to properly remove all weaknesses inherent to the password-based authentication access mechanism:

- By enforcing a mandatory two-step access, SSHLock eliminates the single point of failure identified earlier.
- By disabling password authentication and relying on the strongest cryptography available in OpenSSH, SSHLock removes the threats associated with weaker cryptography (“counter-measure a” above) and eliminates verification, the process by which an attacker can verify the validity of guessed / generated access keys (“counter-measure b” above).
- By providing for a layered-access architecture, including a restricted area between the two access mechanisms (the confinement or honey-pot), SSHLock protects the system's valuable system accounts from being directly exposed to the network.

SSHLock is a transparent solution that upgrades existing systems to bastion hosts virtually impossible to break into, without ever forcing the company's access keys security policy to be modified.

SSHLock is naturally suited for securing servers, though it can also be used as a simple, yet powerful solution for securing remote access to Un*x workstations. To accommodate the latter situation, it may prove practical to relax some of SSHLock's most restrictive hardening constraints by re-enabling system logon through the system's physical terminal, possibly using OPIE or Kerberos authentication.

SSHLock as a valuable alternative to standard access mechanism and SSH service.

By bringing a wealth of security features that, together, provide for a much safer access to computers over the standard password-based access mechanism, SSHLock is a safer, secure and powerful alternative to any password-based access mechanism, including standard SSH services for remote access.

Indeed, the credentials needed to successfully access an SSHLock-protected system, namely two strong passphrase-protected cryptographic key-pairs, knowledge of the TCP/IP ports, sshlock account and command proxy used the SSHLock deployment, name of the enabled (remote) local accounts, are orders of magnitude stronger than that of the standard password-based authentication access mechanism – often a simple, weak password.

However, this extra security does not have to come at a cost. Indeed, SSHLock provides the end-user with a simple, yet powerful interface, comparable in its ease of use to that of SSH, that makes it a breeze to connect to an SSHLock-protected system, thus hiding the complexities of manually passing thru an sshlock, an obvious benefit given the wealth of extra security brought up by SSHLock compared to a single layer SSH access.

SSHLock has been our private solution of choice for many years. Having professionally deployed it on a number of sensitive machines, mainly servers, it has consistently proven to be highly successful in hardening access to them without ever hindering using them. The SSHLock software is accessible at <http://www.franckys.com/products/sshlock/index.html>

Further work

We are currently investigating promising technologies that could be used to provide extra layers of protection around SSHLock and shield computers even more against the danger of the exterior world:

- Port-knocking. Port-knocking is like going to a close bank and knocking on the outside door to politely request that the security access lock be activated so you can enter the bank and access your safe. Adding port-knocking to SSHLock would allow to hide sshd1, making it an on-demand service accessible only upon request, thus offering an extra layer of security.
- Setting up Tor hidden services for SSHLock would provide an additional layer of encryption and server authentication. Given the configuration would only accept connections from the hidden service or from the LAN, and will not disclose the sshd1 fingerprint, people looking at the external traffic will never know the IP address, and will be unable to scan and target other services running on the server.
- Key storage. We could make good use of “ssh-keygen -o -a \$number” to slow down cracking attempts by iterating many times the hash function [5].

References

- [1] https://en.wikipedia.org/wiki/Brute-force_attack
- [2] [https://en.wikipedia.org/wiki/Social_engineering_\(security\)](https://en.wikipedia.org/wiki/Social_engineering_(security))
- [3] <https://en.wikipedia.org/wiki/Airlock>
- [4] <https://www.openssh.com/>
- [5] <https://stribika.github.io/2015/01/04/secure-secure-shell.html>

About the Author:

Franck PORCHER earned a PhD in Computer Science in Paris, France, where he started his career as a researcher for a decade within the flagship of the national aviation industry.

Wishing to explore more of the world and his own ideas, Franck has since reoriented himself as a startup entrepreneur, university professor, general purpose software engineer and opensource developer for the last twenty years, roaming between Tahiti, for its crystalclear lagoons, Belize, for its equatorial exuberance, British Columbia, for its lush rain forests and wild mountains, and Paris, for its delicious breakfasts.

Franck is now a full-time software product editor and publisher.

You can easily find Franck on LinkedIn
at : <https://bz.linkedin.com/in/franckporcher>

You are welcome to email Franck at : franck.porcher@gmail.com

FREENAS MINI STORAGE APPLIANCE

IT SAVES YOUR LIFE.



HOW IMPORTANT IS YOUR DATA?

Years of family photos. Your entire music and movie collection. Office documents you've put hours of work into. Backups for every computer you own. We ask again, *how important is your data?*

NOW IMAGINE LOSING IT ALL

Losing one bit - that's all it takes. One single bit, and your file is gone.

The worst part? **You won't know until you absolutely need that file again.**



Example of one-bit corruption

THE SOLUTION

The FreeNAS Mini has emerged as the clear choice to save your digital life. **No other NAS in its class offers ECC (error correcting code) memory and ZFS bitrot protection to ensure data always reaches disk without corruption and *never degrades over time.***

No other NAS combines the inherent data integrity and security of the ZFS filesystem with fast on-disk encryption. No other NAS provides comparable power and flexibility. The FreeNAS Mini is, hands-down, the best home and small office storage appliance you can buy on the market. **When it comes to saving your important data, there simply is no other solution.**

The Mini boasts these state-of-the-art features:

- 8-core 2.4GHz Intel® Atom™ processor
- Up to 16TB of storage capacity
- 16GB of ECC memory (with the option to upgrade to 32GB)
- 2 x 1 Gigabit network controllers
- Remote management port (IPMI)
- Tool-less design; hot swappable drive trays
- FreeNAS installed and configured



<http://www.iXsystems.com/mini>



FREENAS CERTIFIED STORAGE



With over six million downloads, FreeNAS is undisputedly *the* most popular storage operating system in the world.

Sure, you could build your own FreeNAS system: research every hardware option, order all the parts, wait for everything to ship and arrive, vent at customer service because it *hasn't*, and finally build it yourself while hoping everything fits - only to install the software and discover that the system you spent *days* agonizing over **isn't even compatible**. Or...

MAKE IT EASY ON YOURSELF

As the sponsors and lead developers of the FreeNAS project, iXsystems has combined over 20 years of hardware experience with our FreeNAS expertise to bring you FreeNAS Certified Storage. **We make it easy to enjoy all the benefits of FreeNAS without the headache of building, setting up, configuring, and supporting it yourself.** As one of the leaders in the storage industry, you know that you're getting the best combination of hardware designed for optimal performance with FreeNAS.

Every FreeNAS server we ship is...

- » Custom built and optimized for your use case
- » Installed, configured, tested, and guaranteed to work out of the box
- » Supported by the Silicon Valley team that designed and built it
- » Backed by a 3 years parts and labor limited warranty

As one of the leaders in the storage industry, you know that you're getting the best combination of hardware designed for optimal performance with FreeNAS. **Contact us today for a FREE Risk Elimination Consultation with one of our FreeNAS experts.** Remember, every purchase directly supports the FreeNAS project so we can continue adding features and improvements to the software for years to come. **And really - why would you buy a FreeNAS server from *anyone* else?**



FreeNAS 1U

- Intel® Xeon® Processor E3-1200v2 Family
- Up to 16TB of storage capacity
- 16GB ECC memory (upgradable to 32GB)
- 2 x 10/100/1000 Gigabit Ethernet controllers
- Redundant power supply

FreeNAS 2U

- 2x Intel® Xeon® Processors E5-2600v2 Family
- Up to 48TB of storage capacity
- 32GB ECC memory (upgradable to 128GB)
- 4 x 1GbE Network interface (Onboard) - (Upgradable to 2 x 10 Gigabit Interface)
- Redundant Power Supply

<http://www.iXsystems.com/storage/freenas-certified-storage/>



The Voicemail Scammers Never Got Past Our OpenBSD Greylisting

by Peter Hansteen

We usually don't see much of the scammy spam and malware, but when we went looking for them, we found a campaign where our OpenBSD greylisting setup was 100% effective in stopping the miscreants' messages.

During August 23rd to August 24th 2016, a spam campaign was executed with what appears to have been a ransomware payload. I had not noticed anything particularly unusual about the bsdly.net and friends setup that morning, but then Xavier Mertens' post at isc.sans.edu Voice Message Notifications Deliver Ransomware* caught my attention in the tweetstream, and I decided to have a look.

The first step was, as always, to grep the spamd logs**, and sure, there were entries with from: addresses of [voicemail@](mailto:voicemail@bsdly.net) in several of the domains my rigs are somehow involved in handling mail for.

But no message from voicemail@bsdly.net had yet reached any mailbox within my reach at that point. However, a colleague checked the quarantine at one of his private mail servers, and found several messages from voicemail@ aimed at users in his domains.

Dissecting a random sample confirmed that the message came with an attachment with a [.wav.zip](#) filename that was actually a somewhat obfuscated bit of javascript, and I take others at their word that this code, if executed on your Microsoft system, would wreak havoc of some sort.

At this point, before I start presenting actual log file evidence, it is probably useful to sketch how the systems here work and interact. The three machines skapet, deliah and portal are all OpenBSD systems that run spamd in greylisting mode, and they sync their spamd data with each other via spamd's own synchronization mechanism.

*<https://isc.sans.edu/forums/diary/Voice+Message+Notifications+Deliver+Ransomware/21397/>

**<http://man.openbsd.org/OpenBSD-current/man8/spamd.8>

All of those machines do greytrapping based on the [bsdlly.net](http://www.bsdlly.net/~peter/traplist.shtml) list of spamtraps*, and skapet has the additional duty of dumping the contents of its greytrapping generated blacklist to a downloadable text file** once per hour. Any message that makes it past spamd is then fed to a real mail server that performs content filtering before handing the messages over to a user's mailbox or, in the case of domains we only do the filtering for, forwards the message to the target domain's mail server.

The results of several rounds of 'grep voicemail \$logfile' over the three spamd machines are collected here***, or with the relatively uninteresting "queueing deletion of ..." messages removed, here****.

From those sources, we can see that there were a total of 386 hosts***** that attempted delivery, to a total of 396 host and target email pairs (annotated here***** in a .csv file with geographic origin according to whois*****).

The interesting part came when I started looking at the mail server logs to see how many had reached the content filtering or had even been passed on in the direction of users' mailboxes.

There were none.

The number of messages purportedly from voicemail@ in any of the domains we handle that made it even to the content filtering stage was 0.

Zero. Not a single one made it through even to content filtering.

That shouldn't have been a surprise.

After all I've spent significant time over the years telling people how effective greylisting is, and that the OpenBSD spamd version is the best of the breed.

*<http://www.bsdlly.net/~peter/traplist.shtml>

**<https://home.nuug.no/~peter/bsdlly.net.traplist>

***<https://home.nuug.no/~peter/voicemail/all-voicemails.txt>

****https://home.nuug.no/~peter/voicemail/all-voicemails_nodelete.txt

*****https://home.nuug.no/~peter/voicemail/voicemail_sender_ip.txt

*****https://home.nuug.no/~peter/voicemail/voicemail_ipfromto_all.csv

*****<http://man.openbsd.org/OpenBSD-current/man1/whois.1>

You could take this episode as a recent data point that you are free to refer to in your own marketing pushes if you're doing serious business involving OpenBSD.

And if you're into those things, you will probably be delighted to learn, if you hadn't figured that out already, that a largish subset of the attempted deliveries were to addresses that were already in our published list* of spamtrap addresses.

That means our miscreants automatically had themselves added to the list of trapped spammer IP addresses as intended.

If you're interested in how this works and why, I would suggest taking a peek at the OpenBSD web site, and, of course, I have a book** out (available at that link and via better bookstores everywhere) that explains those things as well.

Relevant blog posts of mine include Keep smiling, waste spammers' time, Maintaining A Publicly Available Blacklist - Mechanisms And Principles, In The Name Of Sane Email: Setting Up OpenBSD's spamd(8) With Secondary MXes In Play - A Full Recipe and a few others, including the somewhat lengthy Effective Spam and Malware Countermeasures - Network Noise Reduction Using Free Tools . To fully enjoy the experience of what these articles describe, you may want to get hold of your own CD set from the OpenBSD store.***

And again, if you're doing business involving OpenBSD, please head over to the project's donations page**** and use one or more of the methods there to send the developers some much needed cash.

**<http://www.bsdly.net/~peter/traplist.shtml>*

***<https://www.nostarch.com/pf3>*

**** <http://bsdly.blogspot.dk/2013/05/keep-smiling-waste-spammers-time.html>*

<http://bsdly.blogspot.dk/2013/05/keep-smiling-waste-spammers-time.html>

<http://bsdly.blogspot.dk/2013/04/maintaining-publicly-available.html>

<http://bsdly.blogspot.dk/2012/05/in-name-of-sane-email-setting-up-spamd.html>

<http://bsdly.blogspot.dk/2014/02/effective-spam-and-malware.html>

https://www.openbsdstore.com/cgi-bin/live/ecommerce.pl?site=shop_openbsdseurope_com&state=department

*****<http://www.openbsd.org/donations.html>*

In addition to the files directly referenced in this article, some related files are available from this directory.* I'll be happy to answer any reasonable queries related to this material.

Good night and good luck.

Update 2016-08-30: I've been getting questions about the currently active campaign that has *document@* as its sender. The same story there: I see them in the greylist and spamd** logs, no trace whatsoever in later steps. Which means they're not getting anywhere.

Update 2016-09-13: A quick glance at a tail -f'ed spamd log file reveals that today's fake sender of choice is *CreditControl@*. Otherwise, same story as before, no variations. And of course, there may have been dozens I haven't noticed in the meantime.

* <https://home.nuug.no/~peter/voicemail/>

* * <http://man.openbsd.org/OpenBSD-current/man8/spamd.8>



About the Author:

Puffyist, daemon charmer, penguin wrangler. Wrote The Book of PF (3rd ed out now, see <http://www.nostarch.com/pf3>), rants on sanity in IT (lack of) at <http://bsdly.blogspot.com/>. Please read <http://www.bsdly.net/~peter/rentageek.html> before contacting.

<http://bsdly.blogspot.dk/2016/08/the-voicemail-scammers-never-got-past.html>

Promote what you are doing with FreeBSD, and what you like and don't like.

Interview with Deb Goodkin, Executive Director for the FreeBSD Foundation

by Marta Ziemianowicz, Marta Strzelec & Marta Sienicka

[BSD Magazine]: Hello Deb, how have you been doing? Can you introduce yourself to our readers?

[Deb Goodkin]: Hi Marta, thank you for taking the time to talk to me about the work we are doing at the Foundation.

I'm the Executive Director for the FreeBSD Foundation. I've been with the Foundation for over 11 years now.

[BSD Mag]: What is the FreeBSD Foundation working on? What is your role in the Foundation?

[DG]: There are a lot of different areas that we are focusing on to help the FreeBSD Project and Community. The main areas that we support are: improvements to the operating system, improving FreeBSD infrastructure, providing legal support, full-time release engineering support, FreeBSD advocacy and education, and facilitating face-to-face opportunities by sponsoring conferences, summits, and other events.

I have overall strategic and operational responsibility for the staff, programs, and expansion, and execution of our mission and vision.

[BSD Mag]: What is your background and why have you decided to join the Foundation?

[DG]: I spent over 20 years in the data storage industry, starting out in research and development of 14" to 2.5" disk-drives. I've worked as a firmware engineer, logic designer, applications engineer, as well as management, and technical sales and marketing roles. I also worked as a consultant for many years doing various technical jobs for storage companies.

I joined the Foundation because the position provided new challenges and the opportunity to use my technical background in a different industry for me.

[BSD Mag]: What is the most important project you are working on at the moment?

[DG]: That is a very difficult question to answer, because we have many projects that are critical to our mission of supporting FreeBSD. So, I'll pick three to focus on here:

1) Accelerating OS improvements. We do this in two ways, first by having full-time staff members to maintain and improve critical kernel subsystems, add features and functionality, and fix problems. Second, by funding and overseeing focused development projects like the recent arm64 port project.

2) Increasing FreeBSD users and contributors by providing FreeBSD advocacy and education around the world. We are doing this by increasing the FreeBSD presence at more open source and technical conferences, creating more informational handouts on topics such as Research on FreeBSD and the Google Summer of Code program, and increasing FreeBSD education in universities and schools.

3) Building and improving the FreeBSD infrastructure and developer tools. We are doing this by helping to improve automation and integration of tools, and adding support for more modern tools to make it easier for new people to get onboard and contribute to the Project.

[BSD Mag]: How did you get interested in FreeBSD? Or actually are you interested in open source software in general? Or are you more interested in the management part of the non-profit corporation?

[DG]: Those are all great questions! I wasn't very knowledgeable about open source before I started working at the Foundation. My background was more hardware/firmware, working on proprietary products. But, once I got involved in this project, I realized what an amazing community this was. People were welcoming to me as a newbie, even though many didn't know my background.

What I love about the FreeBSD Project is that you can learn about software design and architecture from looking at the actual code. FreeBSD has a long history of well thought out and designed code. Anyone can look at this code, try running or experimenting with it, and even contribute changes to it, while gaining marketable skills to get a great job. You also don't need to be a software developer to contribute. The Project offers many ways to gain real-world experience by helping with documentation, participating on teams like ports and release engineering, or playing around with FreeBSD by installing it on a computer, installing helpful software packages, and using that as a platform for many educational opportunities.

[BSD Mag]: I've heard from one of my friends, who is a programmer, that he doesn't like open source mostly because of this fuss about it or around it. I guess he isn't a huge fan of the open source communities either. Do you have any thoughts about it? And is it like that in general, people are divided into two groups: those who like open source and support it and those who don't?

[DG]: There are many advantages of working on an open source project. A person may get involved because of the community, and each one is different. It's important to look at not only the open source product, but what is the community like? What is their philosophy? How are contributions accepted, and how are contributors treated? One of the major advantages of working on an open source project is that you can work in the area that you are interested in, gain new skills and experiences, and showcase your work for potential employers.

[BSD Mag]: I spotted you after receiving a newsletter calling for donations. How is it going this year? What is the money from donations spent on?

[DG]: I'm glad to hear that you are on our mailing list! As of today, Dec 17, we've raised just over \$900,000 towards our goal of \$1,250,000. This is our busiest time of year, when we focus on not only meeting and hopefully exceeding our fundraising goals, but to ask more individuals to make a donation. We also have a goal of having over 2000 donors this year.

We are funded 100% from donations. That means we use the donations to support everything we do, like the projects I mentioned above, as well as sponsoring BSD-related conferences, providing FreeBSD advocacy and education, providing legal support for the project, full-time release engineering support, providing face-to-face opportunities for FreeBSD contributors to work together and with vendors, and to cover our administrative expenses.

[BSD Mag]: Is it difficult to raise the needed amount of money each year? Why should the community get financially involved?

[DG]: I have to admit that it is difficult to raise the needed amount of money each year. We are working on improving our fundraising and business models to increase the funding stream, so we can continue and increase our efforts of supporting FreeBSD. The difficulty lies in getting companies, or the right people at companies, to not only recognize the value of FreeBSD, because I believe they do, but to think about giving back financially and investing in the future of FreeBSD.

Showing strong community support, financially, is a testament to the community behind FreeBSD. It shows people outside the community that this is a relevant operating system with a vibrant and engaged community.

[BSD Mag]: Let's say that FreeBSD enthusiasts think that there are many companies, who use FreeBSD and they should become sponsors. What would be your message to the users, who don't see the point in supporting the project? In 3-5 sentences.

[DG]: First, we have a great template created by a FreeBSD enthusiast who uses it at his company to get them to donate every year. He was happy to share it with us for others to use. <https://www.freebsdoundation.org/donate/donation-letter/>

As a 501c3 non-profit 100% committed to making FreeBSD the best platform for research, computing, products, and education, we need the support of every single user to accomplish our goals. Your donation allows us to increase our efforts toward accelerating operating system improvements and supporting release engineering work leading to timely and reliable releases. You will help us recruit and educate people about FreeBSD. You will allow us to provide more opportunities for companies to work directly with developers and for developers to work together on projects, learn new ideas, and get more work completed. Your investment in FreeBSD today, will keep it the relevant, reliable, and secure operating system that you depend on.

[BSD Mag]: What happens if you don't raise enough money?

[DG]: We ran into this issue this year, with our lower than planned fundraising efforts in 2015. We have investments that we keep for situations like this. When I created the 2016 budget last year, I proposed to the board that we continue supporting the same efforts that we did the previous year, believing this was critical to the project and that potential donors would recognize how important our work is. We are also researching other business models to help with the funding stream, so we aren't 100% reliant on donations.

[BSD Mag]: What do you suggest to a BSD user to keep up-to-date with the whole community? Do you recommend attending conferences, reading papers, books, manuals, or something different?

[DG]: Attending a BSD-related conference is the most productive way to keep up-to-date with what's going on in the BSD world. It's a great way to get energized to contribute more to the Project. We do offer travel grants to FreeBSD contributors who need financial assistance to attend these conferences. Our FreeBSD Journal provides a lot of great articles and information on features, new technologies, and other areas of FreeBSD that also applies to the other BSDs.

I also think the FreeBSD handbook is one of the best ways to learn about FreeBSD.

Lastly, I'd recommend joining a local user group, and if there isn't one, consider starting your own. It's a great way to talk to others who are interested in Unix in general, and to get new ideas in areas that interest you.

[BSD Mag]: What are the biggest challenges you, as an Executive Director of a non-profit organization, have to face?

[DG]: The biggest challenge is getting the funding we need so we can add more resources to help us support more areas of the Project. Though our focus is on supporting FreeBSD, I have to also think about running a company and implementing things that will make us more efficient, stay compliant as a public 501c3, and make sure our employees are taken care of. I don't have enough hours in the day to get to everything.

[BSD Mag]: What are your plans for the future. And Foundation plans?

[DG]: Hopefully, I can continue my work with the Foundation for a long time. With the appropriate funding, I will be able to bring on more resources to help with our efforts. I have some personal interests within the community that I'd like to focus on, including getting more women involved in FreeBSD, and teaching more young people about FreeBSD.

For the Foundation, we have other areas that we'd like to support besides the areas I discussed earlier. Those key areas include improving the developer infrastructure, improving the new user experience, providing training for companies, and increasing our advocacy and education efforts.

[BSD Mag]: Do you have any piece of advice for our readers?

[DG]: Promote what you are doing with FreeBSD, and what you like and don't like. We want to hear these stories and find out what people think. If you are new to FreeBSD, find an old computer and install FreeBSD. Then install packages and see what works and what doesn't. Submit a question to a mailing list or one of the forums. People in the community love helping new people who show a genuine interest in learning.

Get involved! It's a great community with lots of opportunities to contribute, learn, and meet incredibly smart, but approachable people!

I'd like to ask your readers to please donate today to help us continue and increase our support to the FreeBSD Project and Community. Go here to make a donation:

<https://www.freebsdoundation.org/donate/>

Check us out on FaceBook <https://www.facebook.com/FreeBSDFoundation/> and Twitter

<https://twitter.com/freebsdoundation>.

[BSD Mag]: Thank you for finding time for us! Good luck with reaching your goal :)



About Deb:

DEB GOODKIN is the Executive Director of the FreeBSD Foundation. She's thrilled to be in her 12th year at the Foundation and is proud of her hard-working and dedicated team. She spent over 20 years in the data storage industry in engineering development, technical sales, and technical marketing. When not working, you'll find her on her road or mountain bike, running, hiking with her dogs, skiing the slopes of Colorado, or reading a good book.

To find out more about the work we are doing and how we can help you go to:
www.freebsd.foundation.org

Editor's note: Just before releasing this issue, Phoronix has published a happy update: (Written by Michael Larabel in BSD on 28 December 2016)

FreeBSD Foundation Receives Another \$500,000 USD Gift

FreeBSD is ending 2016 on a high note by receiving another "Uranium Level" donation, marking it as an additional \$500,000 USD for their foundation.

Earlier this month the FreeBSD Foundation received a \$500,000 donation from the founder of WhatsApp, Jan Koum. That's on top of Koum giving one million dollars to FreeBSD back in 2014.

Announced today was another \$500,000 USD donation. Though this time the big ticket donor is remaining anonymous.

What they call as a "Uranium level" donation is \$250,000 USD or more. The only two donors at this level were Jan Koum's \$500k donation and this anonymous donation. A list of FreeBSD's 2016 donors can be found here. This year the foundation received \$1,509,493 of their \$1,200,000 goal.

Source: http://www.phoronix.com/scan.php?page=news_item&px=FreeBSD-Uranium-Anonymous

I owe my whole life to the open source movement

Interview with Shawn Webb, Founder of HardenedBSD Project

by Marta Ziemianowicz, Marta Strzelec & Marta Sienicka

[BSD Magazine]: Hello Shawn, how have you been doing? I guess everybody knows you, but can you tell us something about yourself?

[Shawn Webb]: I'm doing great. Thank you very much for asking and for the opportunity to do this interview. We just bought our first home this year. I just did my first DIY project by wiring the house with CAT6.

[BSD Mag]: HardenedBSD. How has it started and why? Where did the idea of creating it come from?

[SW]: HardenedBSD unofficially started three years ago when Oliver and I started working on Address Space Layout Randomization (ASLR) together. At the time, our codebases were separate. I was working on -CURRENT and he was working on -STABLE, causing merge conflicts whenever we would try to merge our code together. We created HardenedBSD so that we could better coordinate development efforts.

[BSD Mag]: What about the name? Where did that come from?

[SW]: Oliver chose it. I like it and it stuck.

[BSD Mag]: What was the most difficult part when creating HardenedBSD?

[SW]: Managing a project that contains tens of millions of lines of code. There's a lot of moving pieces and some people assume someone like the cofounder knows every inch of the code.

We at HardenedBSD have also really tried to maintain a positive community, focused and centered on helping each other out. We don't encourage elitism or arrogance in our community. Sometimes personalities can clash, and we have to wade through that. I'm glad we've sculpted the community in such a way that it's focused on the technical merits of information security and exploit mitigation.

[BSD Mag]: So you are mostly a security guy... But it looks like you do everything for HardenedBSD project: financials, management, PR...

[SW]: That's what happens when you're a two-man team. We have a few other developers (heya Bernard!), but only Oliver and I are in charge of running HardenedBSD.

[BSD Mag]: It means that you have an amazing experience! As a security guy, what's the most difficult part in managing HardenedBSD? Financials or PR? Or something else?

[SW]: All the moving pieces. Sometimes, when I'd rather be writing code, I have to upgrade infrastructure, or spend my time babysitting a long-running process. Juggling the finances, too, is really difficult. If donations are slim, then paying for HardenedBSD comes out of my own personal pocket. Not only do we have recurring hosting costs, but we have to also replace aging hardware. Doing three to four package builds (building 26,500+ packages) per week is rather costly on hard drives.

[BSD Mag]: And you have started from FreeBSD, right? Do you have any other favorite open source system?

[SW]: Yeah. Originally, HardenedBSD was meant simply to be a staging area for our code until we felt it to be good enough to merge upstream to FreeBSD (similar to TrustedBSD in that respect). But now, we're a fork that syncs with upstream FreeBSD every six hours (so, in a sense, a "spork."). I like seeing the work OpenBSD does and I draw inspiration from a lot of what they do.

[BSD Mag]: How is it going with donations this year? I've seen that FreeBSD Foundation isn't even half way to the goal and it's already the end of November...

[SW]: We're grateful for the donations we've received. They've been put to very good use. We've been able to help the FreeBSD community with their efforts to support the RPI3 and aarch64 development in general. With that said, we're running really slim on donations this year.

[BSD Mag]: HardenedBSD is far from reaching the goal, too, but it's so much lower than the other BSD projects! What do you need the money for?

[SW]: Mainly hosting. Running HardenedBSD costs around \$400 per month. If donations don't come in, it comes out of my personal pocket.

[BSD Mag]: If anyone would like to contribute still, how can they do that?

[SW]: We'd love people to give HardenedBSD a try. File bug reports, submit patches, or donate money. We're grateful for all the contributions we've had to date. We look forward to working with the community.

[BSD Mag]: You are also a Security Engineer at G2. Does work on open source projects bring you any income or do you still need a regular job?

[SW]: Work on open source projects brings me no income. I love where I work and they've been extremely helpful in supporting HardenedBSD. Over the past two years, they've donated five servers, including hosting four of them.

[BSD Mag]: You have recently joined OPNsense Core Team. Tell us something about it, how did that happen, why OPNsense, how is it...?

[SW]: I love to eat my own dog food. If I run a firewall that's based on FreeBSD, I'd like to see HardenedBSD in action. OPNsense is an open source firewall, based on FreeBSD. I joined the OPNsense core team to help them import a good portion of the HardenedBSD bits into it. They really loved the idea. I'm grateful that they have received my contributions with open arms.

[BSD Mag]: What about feedback from the open source community? Do you rely on it?

[SW]: We definitely do. I'm always open to productive dialogue. I love talking tech with the community and learning from others.

[BSD Mag]: BSD Mag has already done an interview with you. I've published it on our blog and posted on the magazine's social media. We have received many different comments about HardenedBSD and yourself. Do you have any comment about that? Do you find yourself controversial?

[SW]: "It is not the critic who counts; not the man who points out how the strong man stumbles, or where the doer of deeds could have done them better.

The credit belongs to the man who is actually in the arena, whose face is marred by dust and sweat and blood; who strives valiantly; who errs, who comes short again and again, because there is no effort without error and shortcoming; but who does actually strive to do the deeds; who knows great enthusiasms, the great devotions; who spends himself in a worthy cause; who at the best knows in the end the triumph of high achievement, and who at the worst, if he fails, at least fails while daring greatly, so that his place shall never be with those cold and timid souls who neither know victory nor defeat." – Theodore Roosevelt

[BSD Mag]: Do you have any thoughts about the open source community? What is great, what could be done better?

[SW]: I owe my whole life to the open source movement in general. The career I have, the house I enjoy, the wife I love. Everything in my life is due to people sharing their code and letting me learn from them.

[BSD Mag]: Are there any challenges HardenedBSD is facing at the moment?

[SW]: We're receiving a few contributions here and there. We'd love it if we could establish a bigger developer community. In the near term, we also will be looking for people to mirror our installers and our package repo.

[BSD Mag]: How would you encourage people to join HardenedBSD community?

[SW]: If you're interested in running a system that raises the bar for attackers, give us a shot. We don't claim that any single exploit mitigation is the end-all-be-all of security, but we do claim that having a holistic outlook, adding layers of security an attacker must peel back before seeing success, definitely is worth the time to investigate.

We also recognize that some exploit mitigations depend on others being implemented first. HardenedBSD is the first operating system to ship with SafeStack. However, in order to be effective, SafeStack requires Address Space Layout Randomization (ASLR) at a minimum. SafeStack can be made more effective with W^X, which HardenedBSD satisfies with its PaX NOEXEC implementation.

If both Google and Copperhead feel grsecurity is worth putting into Android, how much more important is it that we have a BSD working on importing grsecurity? HardenedBSD is that BSD. We're incorporating into HardenedBSD the strengths of the PaX and grsecurity patchsets.

[BSD Mag]: What are your plans for the future?

[SW]: I've got quite a few, so I'll give you the highlights: port over our PaX NOEXEC implementation to OPNsense, finish the HardenedBSD Handbook, revamp our PaX SEGVGUARD implementation, import secadm into base, and implement grsecurity RBAC.

[BSD Mag]: Do you have any piece of advice for our readers?

[SW]: Do what you love and take Teddy's advice.



About Shawn:

Shawn Webb is the cofounder of HardenedBSD. He has around a decade of infosec experience. As a former core developer for ClamAV and fan of FreeBSD, he loves open source. Shawn is a member of the OPNsense core team, responsible for porting over HardenedBSD's many features to OPNsense.

We know our stuff well and use the knowledge to truly work FOR our customers

Interview with Henning Brauer, CEO of BS Web Services GmbH

by Marta Ziemianowicz, Marta Strzelec & Marta Sienicka

[BSD Magazine]: Hello Henning, how have you been doing? Can you introduce yourself to our readers?

[Henning Brauer]: I'm Henning, 38, living in Sternschanze, Hamburg, Europe. I've been into OpenBSD as a developer for 14 years now, hope to make at least another 14, > and am involved with several related community activities. Aside from computer stuff, I'm politically active in the neighborhood, and I like table soccer with friends and beer. When injuries allow I also like to ride my mountain bikes.

One missing:

I also like hiking a lot, with Canada and New Zealand being my favorites.

[BSD Mag]: Can you tell us something about your company, BS Web Services GmbH?

[HB]: I started working in a self-employed fashion on computer stuff in 1996, while still being in school. Things changed to what later became BSWS in 1998. We're offering all kinds of hosting services, with a strong focus on security. We strive to provide rock solid service without the hype and marketing cr.., ahem, stuff. Our customers are typically companies renting servers or entire server farms, and/or VMs, often managed by us. We also have a very sophisticated mail system, so we run a lot of email services for a lot of companies, including some very well known entities. We also do shared hosting and domain registrations, etc., of course. We're one of so far pretty few ISPs offering full DNSSEC, with automatic key rotation.

[BSD Mag]: You have been a CEO in your company for more than 20 years. That's impressive! What is the secret?

[HB]: Not sure there is a secret, BSWS didn't grow Google or Facebook style ;)

We've always been very very honest with our customers, truly working for them, and not primarily for our own good. Not trying to cover up anything, in the rare event that something went wrong we've been open about it. We know our stuff - basically, what we don't truly grok, we don't offer. We don't run anything where we didn't at least have a look at the source.

[BSD Mag]: You have a big portfolio of products. Which ones are your flagship ones? Are they all based on open source systems?

[HB]: Everything is either open source or our own code.

Where we really stand out is DNS, mail, and dedicated servers. Our DNS is very sophisticated yet very easy to use for our customers, last, but not least, we're freeing them from dealing with all the complexity in DNSSEC, we support newer mechanisms like TLSA (DANE), S/MIME certs DNS (SMIMEA), the similar for PGP (OPENPGPKEY) should follow soon (there's a nasty bug somewhere in a 3rd party module we need to fully track down).

Our mail system is very, very powerful, with a lot of anti-spam measures. We do support DMARC, have a couple of sender-behaviour heuristics to tell mail servers and spam bots apart that work really, really well, content scanning for spam and, if desired, virus scans as well - the content-scanning techniques are all optional, we generally provide pretty fine-grained control, some down to individual mailbox level. Our system is designed to never lose mail and delays seldom exceed one second; exception being mass mailings of course - that are automatically shifted to special instances to not delay regular mail. Our dedicated server offerings often end up being a managed service - as in, we operate and manage the machines, take care of backups, redundancy, load balancing if needed, etc., etc., etc., so that customers can concentrate on their business and we make sure they have a truly secure and high performance platform to do so.

It's kinda obvious from our use of OpenBSD that security is crucial to us and we really know what we're doing there. We've never ever had any incidents like hacked servers - the worst incidents, besides DDoS, were some horribly insecure webapps installed by customers, either allowing content to be modified or abused for spam sending. But even these are very rare. While there is no bulletproof solution with customers still being able to upload their or 3rd party webapps, measures in our infrastructure make abuse much harder and easy/quick to get detected.

[BSD Mag]: Let's talk about DDoS, as it seems to be all the rage these days. Do you think there will be something more at our disposal than mitigation any time soon? Do you think we need more than mitigation?

[HB]: The DDoS-Situation is out of hand. The access providers are - with only few exceptions - careless and utterly irresponsible. Many don't even implement BCP38 (that's the most basic spoof protection, make sure only packets with one of your IPs as source leave your

network). The big carriers in the middle aren't any better, most now refuse to do anything but blackholing, making the DDoS 100% successful. That's where change needs to happen. Access ISPs need to start showing responsibility, implement BCP38, and they really need some heuristics to detect hosts that are part of a botnet. It is actually getting even worse, the giant 600+ GBit/s DDoS last week was largely carried out by Internet of Terror devices. If there is no improvement, we'll end up with just a few giant ISPs left that can offer hosting, directly or through the formerly independent smaller ISPs - only the Googles, Akamai and Cloudflares of this world would have the capacity to handle large DDoS. We're talking tens of thousands of servers, distributed worldwide. If the entire content side of the internet is in the hands of a few giant corporations, we're pretty much doomed - last but not least, they're probably all under US legislation. NSA's dream. That, at the same time, means that the internet self governance failed. That will call governments for action, and we have the fully government controlled internet.

https://2016.eurobsdcon.org/PresentationSlides/GertDoering_DDoS+Consequences.pdf

[BSD Mag]: It is not difficult to notice that OpenBSD is your favorite one. Any reason why just OpenBSD?

[HB]: The historic reason that brought me to OpenBSD was a Linux server responding extremely poorly to a DDoS - that was in '98 or '99. Looked at all the major BSDs, liked OpenBSD most, stayed. Later became a developer.

But that is not really the story. While that brought me to OpenBSD, there must be a reason to stick with it for so long. It's actually multiple. OpenBSD's proactive approach to security helps enormously in a hostile environment - as an ISP, pretty much every service is public, the attack surface is huge. Using OpenBSD lets us sleep at night instead of fixing hacked servers, etc., etc., etc. But there's still more. For our uses, OpenBSD has been absolutely rock-solid, for ages now. Sensible defaults save us work.

There's another aspect. I obviously know OpenBSD quite well. If something doesn't work the way it should, I can work on fixing it in OpenBSD, or I can poke the developer who wrote the code in question. The so incredibly common "dunno, just reboot" just doesn't exist here; we don't just install something and then offer it as a service, we know our stuff really, really well - or we don't offer it.

[BSD Mag]: You have been very active in the community. Are you still a contributor and active member of EuroBSDcon Foundation?

[HB]: I am actually sitting in the hotel lobby in Belgrade right now, preparing this year's EuroBSDcon, which starts the day after tomorrow. I guess that's a yes :)

I've been on the board of directors pretty much since the foundation existed and have no plans to

change that. This year we have to run the conf without the big local team that organizes almost everything that we used to have, so Philip Paeps and I pretty much function as the local team this year, despite not being local.

[BSD Mag]: What are the major challenges of running the foundation that the community might not realize?

[HB]: There is quite a bit of administrative overhead, last but not least bookkeeping, tax reports and so on. Or foundation is Dutch, but the conf is in a different country every year, that is complicating things quite a bit. Fortunately, that isn't primarily my duty.

Having the conf in another country every year means starting from scratch in many regards, every year. Find a venue, find accommodation, a place to have the social event, transportation, a printer for shirts, bags, badges, etc.; we usually have a local artist for the logo, we need to budget - a lot of that is handled by the local team, which this year was us.

Then there's the program committee - that's more than just selecting from submissions. Keynote speakers have to be found and convinced, the talk selection must be somewhat balanced over the projects, and travel expenses have to be taken into account.

[BSD Mag]: What is the philosophy behind your company?

[HB]: No nonsense!

We're open, we're honest, we don't have a marketing department annoying customers. We know our stuff well and use the knowledge to truly work FOR our customers.

[BSD Mag]: Your slogan “Convince through performance, not through marketing” means that you don't do marketing at all? Why is that? Why do many companies working on and for open source not like to advertise?

[HB]: Well... I personally find marketing extremely annoying, and I'm just not good at it. Nobody on the company really is. Seems like I just don't get along with typical marketing people...

A bit more on the marketing/sales side would be good, actually. Without annoying people though.

[BSD Mag]: Do you believe there is a middle point? Or should marketing be reinvented completely to be a fit to companies like yours?

[HB]: To be quite honest: I have no idea. I really wish I had somebody that took care of all marketing/sales/... in a way that I wouldn't have to care about it at all.

[BSD Mag]: We have been having interviews with companies and contributors for the magazine mostly from USA, Canada, Netherlands, UK and Brazil. How does it look in Germany? Is open source popular? Are there many companies working on and for open source solutions?

[HB]: Our customer base is not limited to Germany at all :)

But yes, Germany is pretty open source friendly. Europe is probably the region where it is easiest to get open source into commercial use. Open source is popular and there are countless companies and individuals working in the field.

[BSD Mag]: What are your plans for the future?

[HB]: We'll continue to be the guys who really know their stuff, we'll continue to work on the internet of the future by both implementing and at times inventing new tech that makes sense and is secure.

Anybody should be able to understand how our digital connected world is working

Interview with Mathilde Ffrench, Founder of echinopsii

by Marta Ziemianowicz, Marta Strzelec & Marta Sienicka

[BSD Magazine]: Hello Mathilde, how have you been doing? Can you introduce yourself to our readers?

[Mathilde Ffrench]: Hi! I'm good :) I'm passionate about Dev&Ops. Basically, I'm a computer science master and have more than 10 years experience from software engineering to systems operations.

[BSD Mag]: Can you tell us something about your company, echinopsii?

[MF]: echinopsii company is a free and open source software company founded to promote our work on the Ariane project. Basically, echinopsii provides yearly support subscriptions on the Ariane project and a dual license system through the Free2Biz license.

[BSD Mag]: What is Ariane, the project you have been recently working on?

[MF]: The Ariane project has been initialized to provide a mapping automation framework for our distributed digital world. The basic idea is to transform the runtime data we can get as operators or provides as developers to ubiquitous browsable map. Then you'll be able to discover and understand your runtime much quicker, improve your compliance, stop upgrading your documentation schemas at any changes, break the IT silos ... and you can imagine much more solutions based on the data we're collecting.

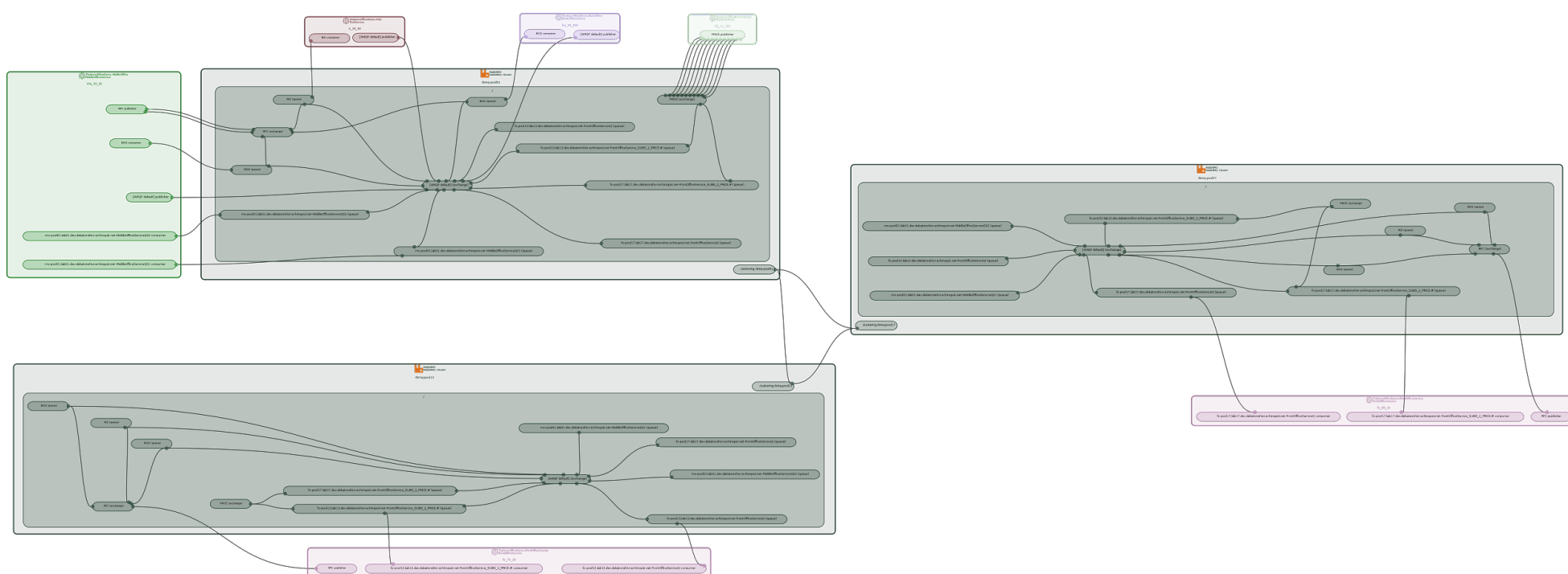
To achieve this huge goal we use a graph database and we augment the graph model to a bi-graph model with a dedicated graph render. Around this main skeleton we provide toolings and APIs to let dev&ops code and provide their own plugins.

Currently, we're providing three different plugins targeting dedicated technology:

- RabbitMQ plugins to map RabbitMQ infrastructures.

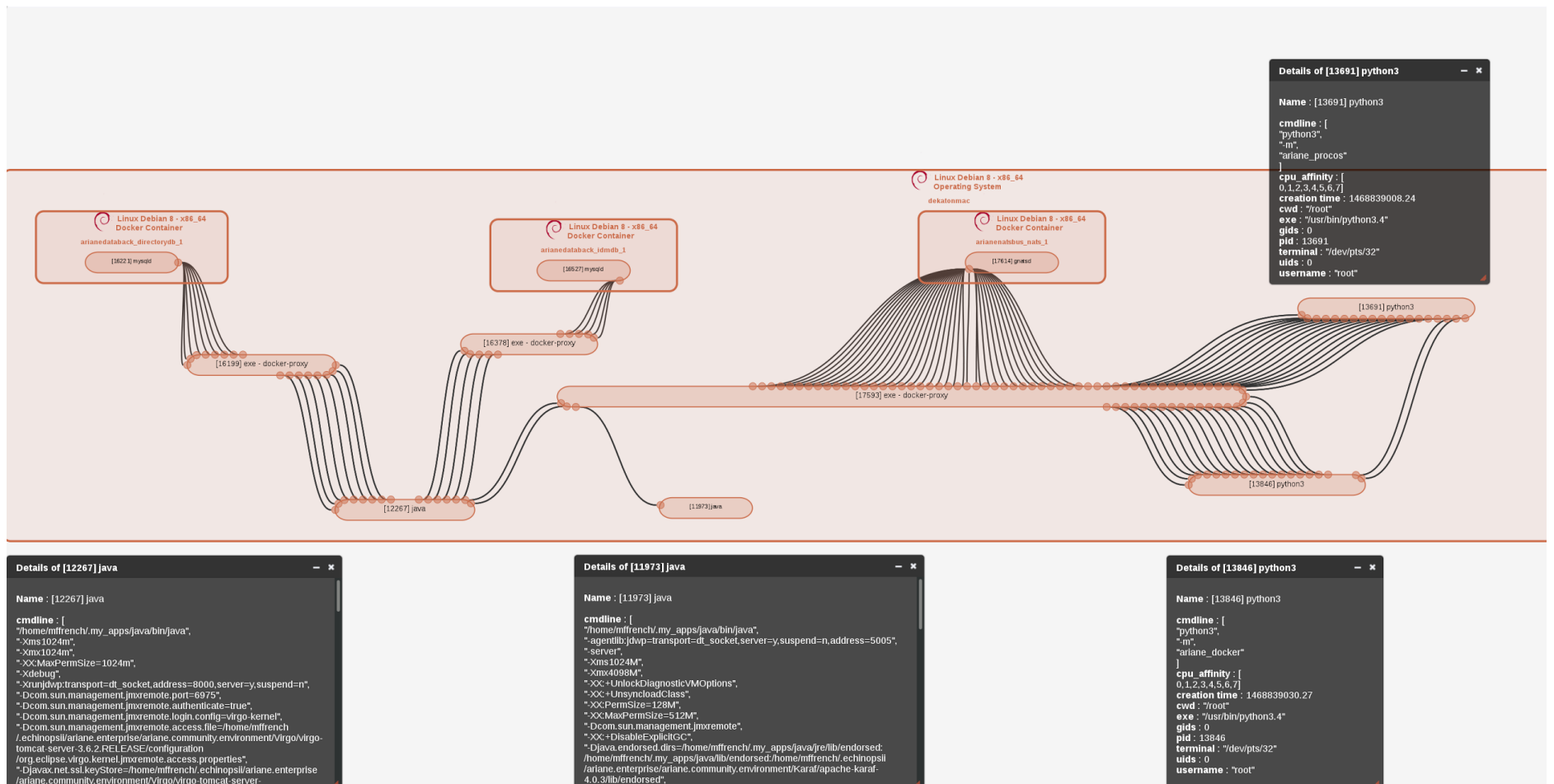
- ProcOS to map your operating system (tested and validated on Linux, OSX and FreeBSD).
- Docker, which augments the ProcOS data to add the docker containers into the map.

In some ways, you can compare Ariane to Prometheus but instead of focusing on time series like in Prometheus, we're focusing on topology and how to unify all this stuff into the same (big) graph database. How is this server connected to remote? How my application is communicating with others through this message oriented middleware? How my actors/threads are working (or not) together? What are the VMs sharing the same hypervisor as mine?



RabbitMQ (AMQP) mapping shows the bigraph model power to provide ubiquitous patterns to map so many things from the digital world. Bigger picture can be found here:

<https://slack-files.com/T04JMETB8-F3DD8F352-f85537a575>



How Ariane is working and deployed with backend DB and plugin can be explained much more easily with automated map! Bigger picture here:

<https://slack-files.com/T04JMETB8-F1X4LJQJK-423434f150>

[BSD Mag]: Which open source systems/programming languages have you been using to create Ariane and why?

[MF]: There are a lot of different languages used in Ariane. First, the core is running on a JRE so we're using Java but also Scala or Groovy. The main reasons I chose Java are:

- I needed a stable and reliable environment with a rich ecosystem where I can peek useful libraries (like akka, hibernate, infinispn, resteasy ...). The Ariane project is a long term project.
- I was able to prototype an ultra modular web application with several OSGI microservices. This was really important as it helped us to speed up prototyping with a robust software architecture and it still helps us today to migrate our monolithics microservices in a more network / containerized microservices way and so improve the Ariane scalability.
- I preferred to embed the Neo4j Graph Database (also done with Java) because at the time, the Neo4J binary protocol didn't exist and so I preferred to manage Neo4J transactions through the Ariane stack and with in memory Neo4J stack calls.

So we're mainly using Java in the Ariane Core. But we're also using Scala for our mapping DSL

So we're mainly using Java in the Ariane Core. But we're also using Scala for our mapping DSL and this usage will grow in the future as Scala provides really cool stuff that can help us improve Ariane and reduce our code base.

Ariane provides a web user interface, thanks to PrimeFaces (based on JSF) and we use Javascript to implement the Ariane map graph render, thanks to RaphaelJS. In the future, we want to migrate all our JSF base to a full javascript UI.

Ariane also provides APIs in Java and Python. We're using Ariane Java API for the Ariane RabbitMQ plugin and Ariane Python API for Ariane ProcOS and Docker plugins. I'd like to deliver a new GO API next year to provide a new NATS plugin.

[BSD Mag]: Do you have a favorite programming language?

[MF]: Not really. In fact, I try to use them depending on the use case and what I want to achieve. If I need to make low level system development, I will use C or Go. Java is great when it comes to building an application like Ariane as it offers a lot of existing libraries. If I need to develop a quick script to monitor my operating system, I will use a scripting language like bash or Python or Ruby... I don't have any church and even I think it's dangerous to have one: it will fatally reduce your perspectives and so your creativity...

[BSD Mag]: Do you have any favorite open source system?

[MF]: I will probably say the system I use everyday: GNU Linux. But I'm curious and a good player so I do not have big preferences and I like when I discover new cool stuff.

[BSD Mag]: What is Free2Biz License?

[MF]: By default, all echinopsii work is licensed with AGPLv3 license. Several reasons why we choose this license:

- The echinopsii's main mission is to provide more transparency on our digital world and for the most people as possible. So it would have been illogical to provide proprietary source code product. We strongly believe that transparency begins with open source code.
- From my experience, open source projects are very often used here in France but companies don't encourage enough contributions. This should be the minimum if you want some viable ecosystem. So we needed a strong copyleft license to build a viable ecosystem around the Ariane project.
- Most of the Ariane microservices are designed to be called from the network and also Neo4J is licensed with GPLv3/AGPLv3 so we chose AGPLv3.

From our point of view, AGPLv3 is OK for all non business use cases like your personal use case or educational use case. But it is definitely not designed for business use cases as you will probably want to use Ariane services from proprietary software, which is avoided by AGPLv3, as it tells you have to publish any derivative code even from remote service calls with AGPLv3.

In France, some guys think people making open source projects don't want to make business. In the echinopsii case, this is exactly the opposite. We need to make business and get money to achieve our goals following end-client use cases and then continue building a viable open source community.

This is the reason why we created the Free2Biz license. The Free2Biz license is a dual license that allows you to use Ariane services from proprietary software. This license is packaged with our professional yearly support subscription, so any professional needs are covered from license to support and maintenance. At the same time, we are delivering exceptions to the AGPLv3 license and we get money from the yearly support/maintenance subscription to make progress on our goals and push back money to the contributors.

[BSD Mag]: What do you think about open source systems and the open source community in general? Have you ever been a part of one of the open source communities?

[MF]: My first contributions on an open source project were on Apache ActiveMQ. At the time, I was working as a software engineer in some company's R&D department but this kind of contribution was an exception and not the rule so the link between the company and the project community was not as close as I would like it to be. Then as an OPS, I had to work with proprietary products almost exclusively and at the end, I was missing the code source - but during this period, the Ariane project idea matured as I needed some tools to help automate reverse engineering. Since I worked on the Ariane project, I'm much closer to the open source projects I use (like Akka, Neo4j or NATS ...) and I'm always happy to help improve Ariane project dependencies. :)

I think open source systems are just as necessary as the air around us to breathe. The main reasons why we're assisting with the current software revolution is thanks to open source and open innovation movements. And finally, our digital security is closely linked to open source system and transparency. Basically, I personally am a bit paranoid when I use most of the proprietary connected devices/software I own because they will at least be able to send some of my data to to their constructors without my consent. I say at least because when I was testing the Ariane Pro-COS plugin on Apple OSX, it displayed a beautiful built-in backdoor and so I've really no trust anymore on this system, for example. In the case of open source software, even if I don't take time to read the underlying source code and I know there is a good probability somebody like me will do the work and watch the stack for me... my paranoia goes a little down and I'm healthier :)

[BSD Mag]: Why open source? Why create a free product?

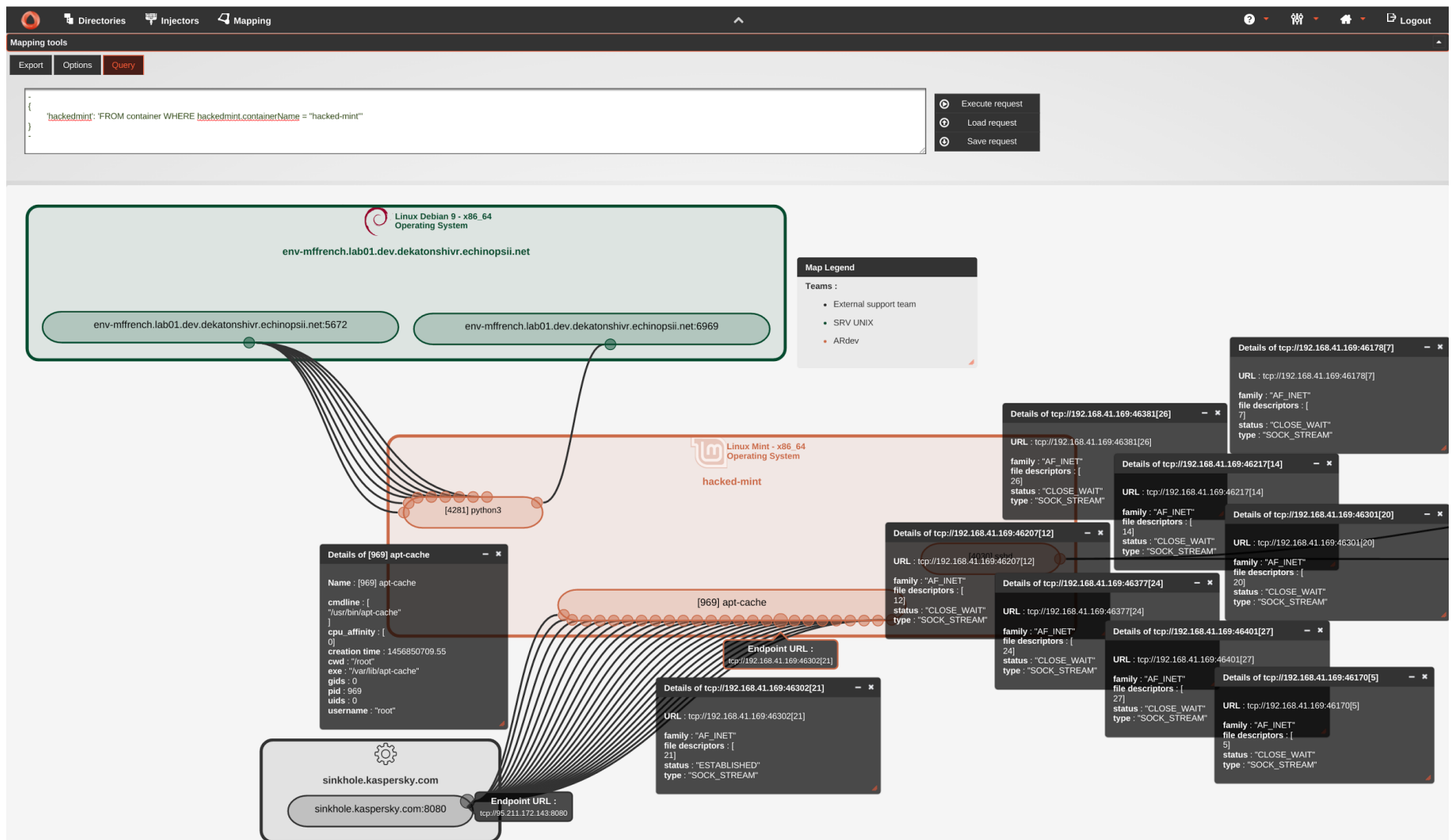
[MF]: As said already, we want to help provide more transparency in the digital world through the Ariane project. Transparency for the digital world begins with open source projects any citizen can look at and modify to fit their needs.

Keep in mind we're not only focusing on big companies with big infrastructure to get a lot of money. We also want to help people to understand how their systems are working because we deeply think getting a clear picture of how your house (full of IoT) is working should be a right like the one you have to breathe. This is the reason Ariane is a long term free project; we want to encourage the people who know things (let's call her/him the contributor) teach the others who don't by providing free Ariane plugins and then complete the map.

Then if the contributor's plugin catches the business interest, the contributors would probably be interested to push this plugin in the Free2Biz license canvas to get back some money as long as she/he supports and maintains it.

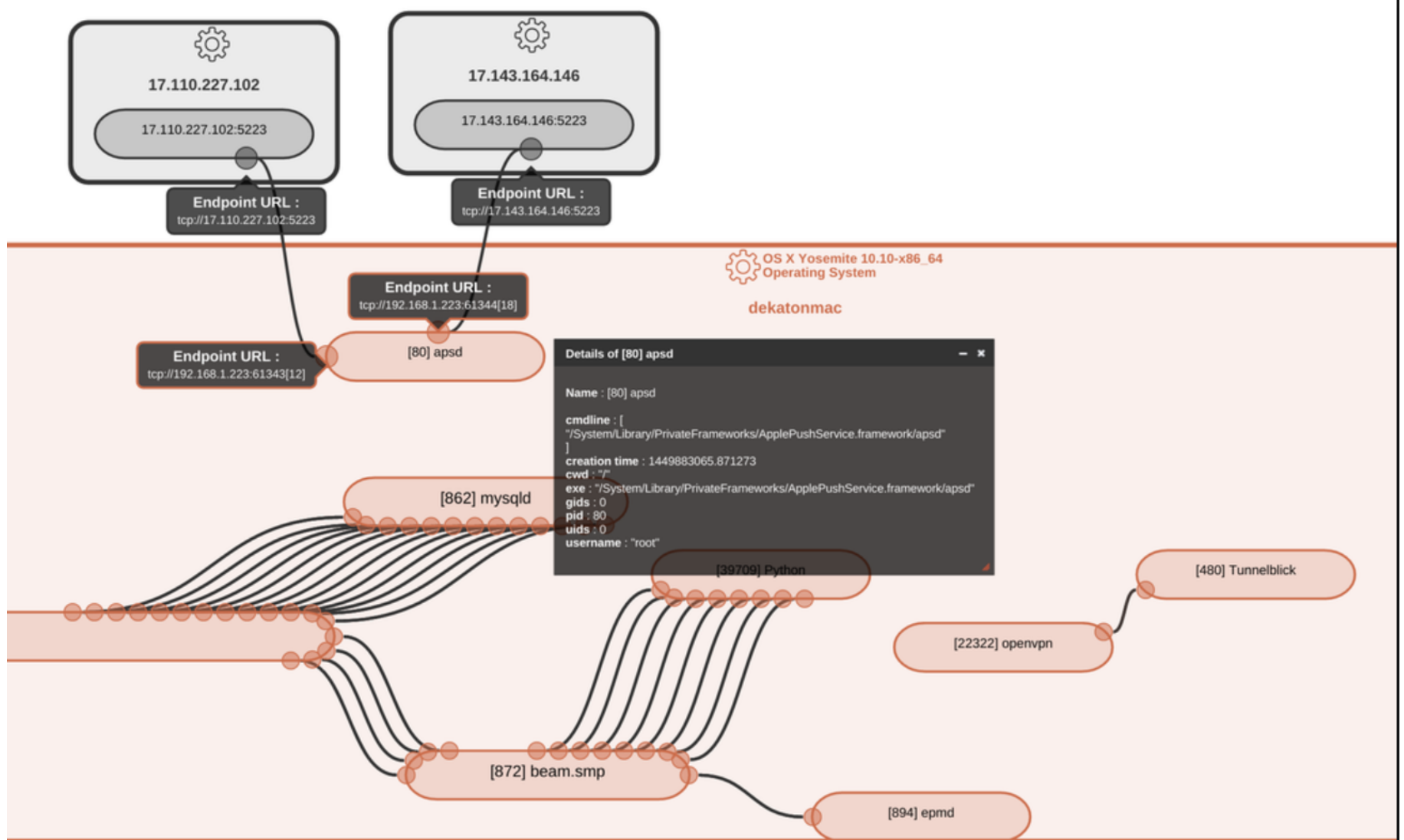
[BSD Mag]: What are you most proud of regarding Ariane development so far?

[MF]: As far as I know, this is the first framework that tends to help you automate your IT mapping and display the part of infrastructure you're interested in (which could be just an operating system running on FreeBSD or something more complex like what's between two points). So I'm proud of the innovation side of this project. I'm also proud to provide a project that could help anybody understand how the digital world is working: I'm already able to explain to my mom what a backdoor is ;)



You can see the backdoor in the hacked Linux Mint in a very easy way on Ariane (a root process connected to external server). Bigger picture can be found here:

<https://slack-files.com/T04JMETB8-F3CKK1AN4-2fbb791182>



echinopsii @echinopsii · 27 déc. 2015

#OSX as a backdoor : #Apple push notification daemon running as root and controlled from the cloud... #GoodToKnow



1



13



3



The same kind of backdoor on OSX. But this time it's a built-in one!

[BSD Mag]: Is there any philosophy behind your company?

[MF]: echinopsii's culture is mainly coming from the Ariane project big idea: reduce the misunderstanding gap between human and computing area by providing an automated map. To get some income, we're focusing on the B2B market to help companies' operations but the long run target is helping lambda people browse their digital home as easy as they are playing on Civilization 6.

The execution path also brings a major echinopsii's culture point: we deeply believe innovation in the digital world is coming from the bottom and so it has an impact on the organization as we need to leverage freedom, creativity and passion.

[BSD Mag]: Is there anything you think everyone should be doing to achieve that?

[MF]: We need to setup horizontal and open minded organizations as much as possible. This is why I think some organization patterns, like holacracy, are very interesting to follow in this case. It

will probably not been enough but I think this is a good starting point.

[BSD Mag]: How is it to be a woman in this very masculine world? More, how is it to be a female founder of a tech company?

[MF]: Not so bad... I don't say I didn't face sexism comments in my career but in some way it helps to filter the people you know you will waste your time with. I don't want to undermine the male/female equality subject, as I see it as a big and complex one and I'm really concerned about it, but the Ariane project is so exciting I prefer keep the focus on it.

[BSD Mag]: What are the challenges your company is facing at the moment?

[MF]: There are two main challenges for echinopsii.

The first one is we're still on the 0 to 1 area. Thanks to the Innov&Connect Incubator program from BNP Paribas and our partner Neo4j, we've got some good feedback from prospects here in France. But I still didn't find the path to transform these prospects into (lean) users and so make progress the project through lean processes.

The second one is making the open source community grow. There are a lot of technical challenges around the Ariane project that could interest many software engineers (from pure system monitoring to the mapping graph render). So we plan to setup a bounty source program next year to help us on this. We're also looking to setup some fair rules to share the Ariane project value between the contributors. If any of your readers are interested in these subjects I'll be happy to get in touch :).

[BSD Mag]: How can people get in touch?

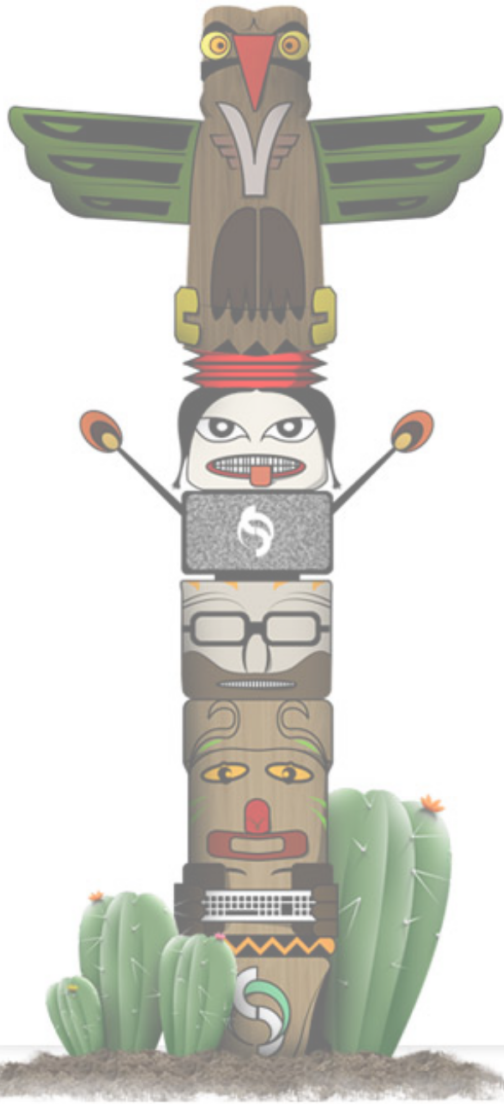
[MF]: You can send me e-mail to mathilde.ffrench@echinopsii.net.

[BSD Mag]: Any plans for the future?

[MF]: On the Ariane project, there are plenty of tasks to improve and complete the current map we have. The prioritization of these tasks will depend on the client's support and new contributors. To give you some idea of these tasks:

- Continue our effort to transform current Ariane OSGI microservices to container ready microservices.
- Add the time dimension to our map to follow the changes over time.
- Improve our mapping graph render to add the shared bigraph display.
- Improve our ProcOS plugin to get system events from the kernel directly and stream them to

the Ariane Mapping service (we are currently pooling every X second and batching to Ariane Mapping service).



- Improve our Docker plugin to add docker overlay network into the map.
- New API: GO
- New plugins: Cassandra, Libvirt, NATS, Rocket, ...

[BSD Mag]: Do you have any piece of advice for our readers?

[MF]: There are several approaches regarding computing. These approaches are well covered in the Robert Milner keynote about ubiquitous computing and I personally follow the Robert Milner vision: “Ubiquitous computing will empower us if we understand it”. I think anybody should be able to understand how our digital connected world is working as we will use it for any services. I think as experts we really need to be careful on the digital world we’re building; we need to promote transparency as much as possible to keep trusting on it.

Freedom comes with Knowledge

About Mathilde:

Born in Lyon France. I’ve a computer science master from ENSIMAG. I’ve been working for several years at Axway as a Software Engineer (R&D dept in Paris) where I had the opportunity to work on enterprise file transfers and message oriented middleware product development (C, Java). Then I moved to a consulting company to get experience in a real production environment (Societe Generale and other clients). Inspired by these experiences, I founded the echinopsii startup with the help of my friends and associates. The main idea of the echinopsii startup is to provide and support a new free and open source framework (the Ariane project), which helps create a detailed and automated map of our digital world and so facilitates and accelerates our understanding of this moving digital world.

DEVOPS WITH CHEF ON FREEBSD

General description:

This training class teaches the tools, best practices and skills to automate your FreeBSD servers. Training will be loaded with practical real world tools and techniques. This training will send you back to work with immediately useful hands on experience to implement Devops in your IT projects.

Course launch date: **1st of December 2016, Self-Paced.**

Duration: 18 hrs.

What will you learn?

- Learn what Devops is and its importance.
- Learn to leverage infrastructure automation using the leading configuration management tool: Chef.
- How it's changing the industry.
- Transform IT from an unpredictable environment to a stable, repeatable and scalable environment.
- Integrating configuration tools into the IT workflow.

What skills will you gain?

- Configure development workstation.
- Understand the Chef architecture.
- Understanding different resources and automation.

Module 1: Introduction to Devops and Chef.

Module 2: Understanding Chef resources.

Module 3: Setup the workstation as a local development environment.

Module 4: Bootstrap a node with Chef server.

Module 5: Get started with writing your first cookbook

Module 6: Recipes, Attributes, Metadata, Templates, etc.

Module 7: Roles, Environment, Search, etc.

Module 8: Using syntax and linting tools like: Foodcritic and Rubocop

Module 9: Writing unit test and integration tests.

There is only 50 seats available, so hurry up, book your seat as soon as possible.

You can find a full description of the course here:

<https://bsdmag.org/product/w05-devops-chef-freebsd/>

If you will have any questions regarding the course, don't hesitate to contact us at

marta.ziemianowicz@bsdmag.org

VMware Storage DRS to the Rescue for Integrating TrueNAS Data Storage

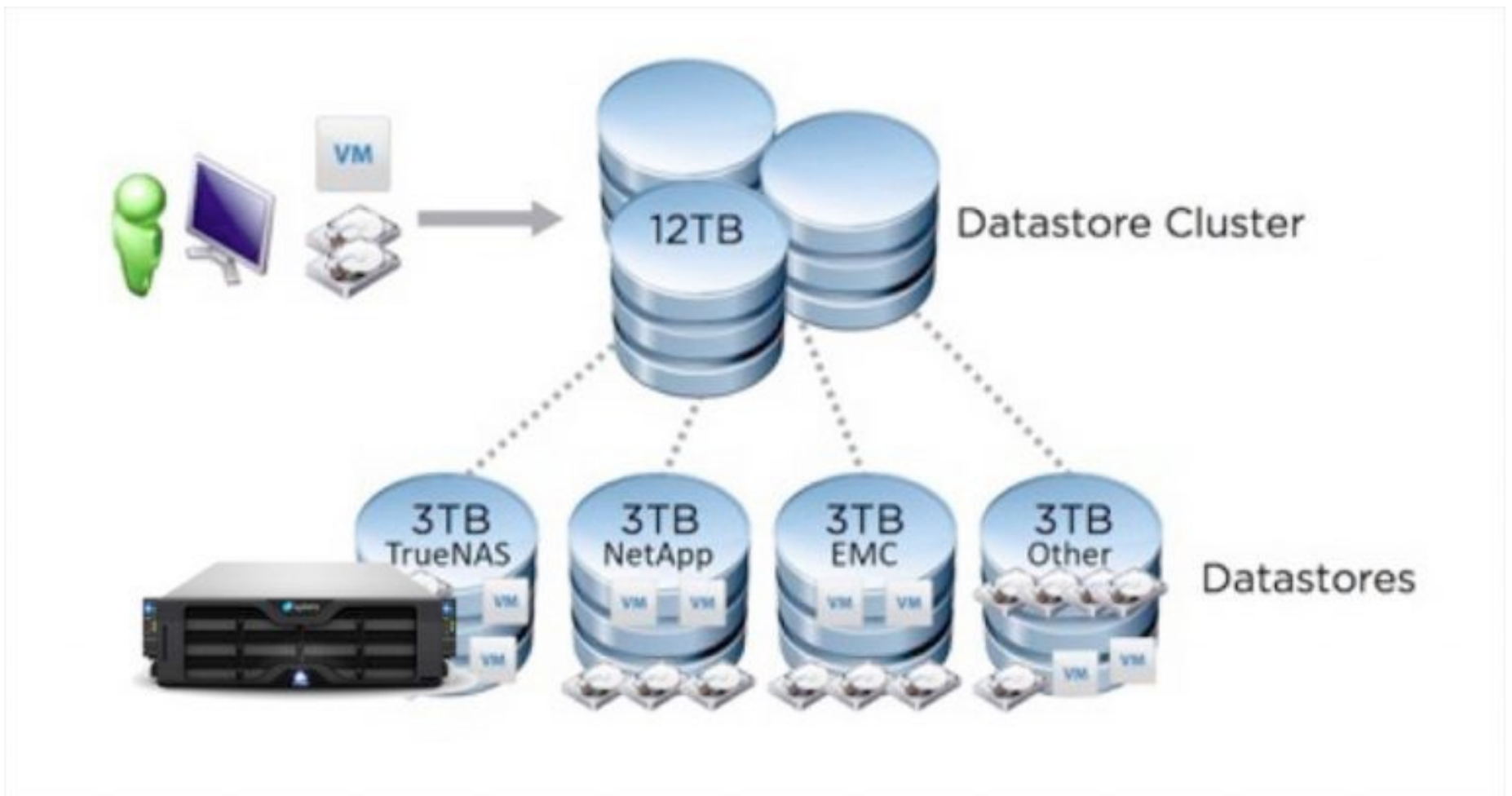
by Brad Meyer, Technical Marketing Engineer, iXsystems

Integrating a new data storage system into an existing VMware environment can challenge even the most experienced administrator. How do you integrate the new storage into VMware while minimizing operational impact? Which VMs should move to the new storage? How do you ensure that you do not overload one storage system over another? How do I retire an older storage system without impacting my VMware environment? How much time does it take to manage these decisions? VMware Storage DRS is the answer.

Introduced with VMware vSphere 5.0, Storage DRS provides smart virtual machine placement and load balancing mechanisms based on I/O and space capacity. It also helps decrease operational effort associated with the provisioning of virtual machines and monitoring of the storage environment [1]. TrueNAS* from iXsystems fully supports and integrates with VMware vSphere Storage DRS.

vSphere Storage DRS offers five key features: resource aggregation, initial placement, load balancing, affinity rules, and datastore maintenance mode. In this article, I will focus on just the resource aggregation and datastore maintenance mode and how TrueNAS from iXsystems integrates into VMware SDRS via storage aggregation.

**<https://www.ixsystems.com/truenas/>*



Resource aggregation is the core component that all other Storage DRS features utilize. Resource aggregation is just that. It is a collection of resources, in this case datastores, clustered into a single unit of storage used by the vCenter Server to allocate storage for VMs. A datastore cluster enables smart and rapid placement of the virtual disk files of a virtual machine and the load balancing of existing workloads. The figure below shows four different 3TB datastores clustered into a single 12TB datastore cluster.

With vSphere Storage DRS 5.1, a vCenter Server can support up to 32 datastores in a single datastore cluster and can have as many as 256 total datastore clusters per vCenter Server instance. vCenter Server requires an Enterprise vSphere license to enable Storage DRS.

The “VMware Ready” certified TrueNAS products from iXsystems provide flexible data storage for VMware via NFS, iSCSI, or FC storage protocols all supporting VMware VAAI. Use a single protocol or use multiple protocols simultaneously to provide datastores that integrate into any Storage DRS cluster. Typical best practice is to provide datastores of similar size and performance characteristics in a single Storage DRS cluster. However, you can mix and match storage together, particularly to migrate from one storage system to another. The only limitation is that you cannot mix block and file protocols in the same datastore cluster.

Storage DRS provides a Datastore Maintenance Mode. In datastore maintenance mode, all registered VMs on that datastore migrate to the other datastores in the datastore cluster. Storage vMotion manages the VM migration under the covers. This extremely valuable feature allows you to integrate a new storage system into your VMware environment and seamlessly migrate all your VMs from your old storage to your new storage. This feature provides a simple, safe, and elegant way to integrate TrueNAS into your VMware environment and retire your old, out-of-date storage system. In automation mode, the migration recommendations generated by datastore maintenance mode are automatically accepted by vCenter and in manual mode are presented to the administrator to validate.

VMware Storage DRS provides a wide range of valuable tools to assist in the overall and ongoing administration of your vSphere environment. Storage DRS enables users to integrate TrueNAS datastores into vCenter Server and move their VMs from an old, out-of-date storage system on to TrueNAS. This is just one example of how Storage DRS can simplify your life as a VMware administrator.

[1]“Understanding VMware vSphere 5.1 Storage DRS”, VMware Corp, March 2013

To learn more about how TrueNAS from iXsystems supports vSphere, contact us at 1-855-GREP-4-IX (855-473-7449) or email us at [**Sales@iXsystems.com**](mailto:Sales@iXsystems.com).

More than 30 years have passed since he first typed at a computer keyboard. Rob Somerville looks back over the major changes in the IT industry and enquires what will be the next revolution on the horizon.

by Rob Somerville

I am getting old. I can't honestly remember the first computer I laid my typing fingers on, if it was a TRS-90, an Atari Amiga, a BBC Micro or a Sinclair ZX Spectrum. At the time I was an impoverished single man living in a bedsit in London, so any computers I managed to gain access to always belonged to other people. The major blessing I had at the time was that I was employed by The London Science Museum, and I had access to lots of electronics, experimental kits and, of course, the odd computer or two. My first professional task was to diagnose a problem with a department's aging computer that ran CP/M. The hard disk was failing, and the joy I experienced diagnosing the fault sealed my fate and I decided that I needed to move from analogue electronics to computing.

The first computer that I owned was a Sinclair Spectrum, purchased from WH Smith after saving diligently for a few months. The keyboard was terrible, and the infernal device had a habit of losing connection with the expansion pack, which would cause the computer to crash when manually typing in any programs in BASIC. Pre-loading programs from a cassette player was also problematic, as the volume on the tape player had to be absolutely spot on (and the heads clean on the

player, as well), otherwise the program would corrupt and the Spectrum would crash with either spectacular screen effects on the attached television or a digital howl of anguish through the single speaker. Eventually, I graduated to an Amstrad PC1640, which not only had a hard disk, but a 360K floppy drive and a colour screen. I was living on baked beans for a few months after the financial impact of purchasing that computer, which was worth over £800 in today's money.

Not long after that, I landed my first full time IT job and with the corresponding increase in salary and status, considered purchasing a genuine IBM 386 clone, but in reality, with the hours I worked, I spent more time in the office than at home. I was becoming a typical IT geek, working late into the night and often at weekends, as deadlines were deadlines. It was during this time I ended up supporting my first 10Base2 thin Ethernet Novell network, and discovered the joys a rock solid network O/S, and a PICK database (Advanced Revelation), although the cabling was vulnerable to failure, especially if the terminators fell off or if the BNC crimps were not properly secured. My next job was with a multi-national as a Novell network administrator, and I was introduced to the joys of Token Ring,

proper Compaq servers, OS/2 and Microsoft Windows.

I left the security of full time regular employment to join the world of the freelance, and aimed to make my fortune. I was now in the fortunate position to be able afford some decent IT kit, and I spend £10K (almost £18K in today's money) on a top of the range 486, a high quality monitor, a flat bed scanner with a hopper feed and an ISDN data connection and the latest US Robotics modem. The first reasonably sized mobile phones were beginning to appear and become affordable, albeit on a rental basis. The Internet was still a way off, and Bulletin Boards and dial up connections were the most common way of engaging with the online community. After a few years, the first major revolution had taken place, and businesses were abandoning my speciality (Novell) for Microsoft desktops and rudimentary workstation to workstation communications. By then I had returned to full time employment in the '90s and by then the next revolution – TCP/IP and the Internet – started to make an impact. Roll on to the middle and late '90s, and the Internet, the World Wide Web and the Dot Com boom was in full swing. What is interesting to note about these times is that each wave of technological revolution was wider and stronger, impacting more and more sectors and the costs were falling dramatically – the democratisation of technology was well under way, and almost everybody was becoming an IT “expert”. By post 2000, the Y2K bug and the bust of the Dot Com bubble, the marketplace had fully transformed from a specialisation to consumer culture.

The next four revolutions - mobile devices, on-line services, social media and Open Source - were clearly driven and underpinned by the latter revolution. The impact this has had on our lives in every sphere has been incalculable, and the strength of the embrace by which the tentacles of technology has grasped our world is clear from the shift towards business continuity, risk management and data security being taken note of, although not as seriously as they probably should be.

The next revolution will be very different from those of the past if we dare move past the critical mass where technology becomes our master rather than our servant. The Internet of Things, Artificial Intelligence, the Cloud, instantly available connectivity anywhere, GPS, drones, driver-less cars, transhumanism, reliable facial and biological recognition were the bread and butter themes of earlier generations of Science Fiction and Dystopian writers. These elements are emerging to become a daily reality, yet the moral, ethical and political framework has not kept pace. We are heading towards a scenario where privacy is a costly privilege and not a right, where more and more decisions are made for us rather than by us, and our lives are becoming almost inseparable from the machine. Like the worst scenes from the movie the Matrix, as human beings, our value is in danger of being evaluated not by our living, real world peers but by an amorphous catalogue of systems to which there is no appeal or rationalisation.

There is hope though. Provided we refuse to allow the worst excesses of technological development to strip us of our individuality and humanity, like the fathers and children of

Open Source, we can have our cake and eat it – for the benefit of everyone. All it will take is a few pioneers, a community of like minded individuals whose vision sees past the inevitable march towards uniformity, a race to the bottom where “one size fits all”. It is clear that the Internet age has raised a generation of media savvy, socially connected netizens, and it is to this generation us older geeks must pass the mantle of responsibility and wisdom.